

Vadim Ermolayev, Heinrich C. Mayr, Mykola Nikitchenko,
Aleksander Spivakovsky, Grygoriy Zholtkevych, Mikhail Zavileysky,
and Vitaliy Kobets (Eds.)

ICT in Education, Research and Industrial Applications: Integration, Harmonization and Knowledge Transfer

Proceedings of the 8-th International Conference ICTERI 2012

Kherson, Ukraine
June, 2012

Copyright © 2012 for the individual papers by the papers' authors.
Copying permitted only for private and academic purposes. This
volume is published and copyrighted by its editors.

Preface

ICTERI is concerned with interrelated topics from infrastructure to education of ICT that are vibrant for both, the research and the industrial community.

With pleasure we present you the selected papers of ICTERI 2012, the eighth edition of the International Conference on Information and Communication Technologies (ICT) in Education, Research, and Industrial Applications: Integration, Harmonization, and Knowledge Transfer, held at Kherson, Ukraine on June 6-10, 2012.

The conference scope was outlined as a constellation of the following themes:

- ICT infrastructures, integration and interoperability
- Machine Intelligence, knowledge engineering (KE), and knowledge management (KM) for ICT
- Cooperation between academia and industry in ICT
- Model-based software system development
- Methodological and didactical aspects of teaching ICT and using ICT in education

Those topics were grouped in two tracks: (1) ICT in research, industrial deployment, and knowledge transfer; and (2) Methodological and didactical aspects of teaching ICT and using ICT in education.

A visit to Google Analytics proves the broad interest in the ICTERI themes. Indeed, between November 15, 2011 and May 15, 2012 we have received almost 3 000 visits to the conference web site from 71 countries (302 cities). Most prominently, the proportion of visits originating from search engine queries was growing constantly – indicating the growing interest to ICTERI 2012.

This year we made ICTERI a more structured event – comprising the main conference, three co-located workshops, and IT talks panel. The main conference program has been composed of top-rated submissions evenly covering all the themes of ICTERI scope – as shown in the tag cloud of ICTERI Key Term use.



The workshops formed the corolla around the main ICTERI by focusing on particular sub-fields relevant to the conference theme. In particular:

- Workshop on Dynamics and Evolution in Intelligent Systems (DEIS) deals mainly with Machine Intelligence, KE, and KM

- The scope of the workshop on Integration of IT in Economics Research (ITER) is more within the topic of cooperation between academia and industry
- Workshop on Algebraic, Logical, and Algorithmic Methods of System Modeling, Specification and Verification (SMSV) focuses on model-based software system development

Traditionally ICTERI hosts also our invited IT Talks panel for industrial speakers who wish to present their cutting edge ICT achievements.

Last but not least, this year's issue of ICTERI closely cooperates with another international conference with a related scope – the 4th International United Information Systems Conference (UNISCON 2012, uniscon.org) held at Crimean State Humanitarian University, Yalta, Ukraine on June 1-3, 2012. UNISCON invited four of the top-rated papers of ICTERI that best match their themes and complement the program. ICTERI also invited three of the best UNISCON papers to be presented in our program. We firmly believe that this exchange made both of our events even better, more balanced and interesting for our attendees.

Overall ICTERI attracted a substantial number of submissions – a total of 70 comprising the main conference and workshops. 48 submissions to the main conference broadly fell into the four genres: full research papers (18), short research papers (20), discussion or problem analysis papers (9), and the papers on industrial experience or case study (1). As for thematic coverage, 25 papers were submitted to Track 1 and 23 – to Track 2.

Out of those 48 submissions we have accepted 25 high quality and most interesting papers to be published in our proceedings. The acceptance rate was therefore 52.08 percent. In addition to those selected publications we included the abstracts of our invited talks. The talk by our keynote speaker Prof. Grigoris Antoniou is on Formal Foundations for RDF Evolution and Repair. It is followed by the talk of Prof. Martin Strecker on Abstraction and Verification of Properties of a Real-Time Java.

The conference would not have been possible without the support of many people. First of all we would like to thank the members of our Program Committee for providing timely and thorough reviews, and also for being cooperative in doing additional review work. We are also very grateful to all the authors who submitted papers to ICTERI and thus demonstrated their interest in the research problems within our scope. We would like also to thank the local organizers of the conference whose devotion and efficiency made ICTERI a very comfortable and effective scientific forum.

June, 2012

Vadim Ermolayev
Heinrich C. Mayr
Mykola Nikitchenko
Aleksander Spivakovsky
Grygoriy Zholtkevych
Mikhail Zavileysky
Vitaliy Kobets

Organization

General Chair

Aleksander Spivakovsky, Kherson State University, Ukraine

Steering Committee

Vadim Ermolayev, Zaporozhye National University, Ukraine

Heinrich C. Mayr, Alpen-Adria-Universität Klagenfurt, Austria

Natalia Morse, National University of Life and Environmental Sciences, Ukraine

Mykola Nikitchenko, Taras Shevchenko National University of Kyiv, Ukraine

Aleksander Spivakovsky, Kherson State University, Ukraine

Mikhail Zavileysky, DataArt, Russian Federation

Grygoriy Zholtkevych, V.N.Karazin Kharkiv National University, Ukraine

Program Co-chairs

Vadim Ermolayev, Zaporozhye National University, Ukraine

Heinrich C. Mayr, Alpen-Adria-Universität Klagenfurt, Klagenfurt, Austria

Natalia Morse, National University of Life and Environmental Sciences, Ukraine

Workshops Chair

Mykola Nikitchenko, Taras Shevchenko National University of Kyiv, Ukraine

Tutorials Chair

Grygoriy Zholtkevych, V.N.Karazin Kharkiv National University, Ukraine

IT Talks Co-chairs

Aleksander Spivakovsky, Kherson State University, Ukraine

Mikhail Zavileysky, DataArt, Russian Federation

Program Committee

Mizal Alobaidi, Tikrit University, Iraq
Costin Badica, University of Craiova, Romania
Tobias Bürger, Capgemini, Germany
Andrey Bulat, Kherson State University, Ukraine
Maxim Davidovsky, Zaporozhye National University, Ukraine
Anatoliy Doroshenko, National Technical University NTU KPI, Ukraine
Louis Feraud, Paul Sabatier University (Toulouse 3), IRIT, France
Jose Manuel Gomez-Perez, Intelligent Software Components (iSOCO) S.A., Spain
Vladimir Gorodetsky, St. Petersburg Institute for Informatics and Automation of the Russian Academy of Science, Russian Federation
Sung-Kook Han, Won Kwang University, Korea
Mitja Jermol, Jožef Stefan Institute, Slovenia
Jason J. Jung, Yeungnam University, Korea
Nataliya Keberle, Zaporozhye National University, Ukraine
Ron S. Kenett, KPA Group, Israel
Christian Kop, Alpen-Adria-Universität Klagenfurt, Austria
Hennadiy Kravtsov, Kherson State University, Ukraine
Vladyslav Kruglyk, Kherson State University, Ukraine
Mikhail Lvov, Kherson State University, Ukraine
Mihhail Matskin, Royal Institute of Technology (KTH), Sweden
Natalia Morse, National University of Life and Environmental Sciences, Ukraine
Julia Neidhardt, Vienna University of Technology, Austria
Andriy Nikolov, Knowledge Media Institute, The Open University, United Kingdom
Aljosa Pasic, ATOS Origin, Spain
Vladimir Peschanenko, Kherson State University, Ukraine
Sergey Rakov, Ukrainian Center for Education Quality Assessment, Ukraine
Kyryl Rukkas, V.N.Karazin Kharkiv National University, Ukraine
Abdel-Badeeh Salem, Ain Shams University Abbasia, Egypt
Wolfgang Schreiner, Research Institute for Symbolic Computation (RISC), Johannes Kepler University, Austria
A.V. Senthil Kumar, Hindustan College of Arts and Science, India
Vladimir A. Shekhovtsov, Alpen-Adria-Universität Klagenfurt, Austria
Oleksandr Sokolov, National Aerospace University "Kharkiv Aviation Institute", Ukraine
Marcus Spies, Ludwig-Maximilians-Universität München, Germany
Martin Strecker, Paul Sabatier University (Toulouse 3), IRIT, France
Vagan Terziyan, University of Jyväskylä, Finland
Marcel Tilly, European Microsoft Innovation Center, Germany
Nikolay Tkachuk, National Technical University "Kharkiv Polytechnic Institute", Ukraine
Yuriy Tryus, Cherkasy State Technological University, Ukraine
Mikhail Ugrumov, National Aerospace University "Kharkiv Aviation Institute", Ukraine
Helmut Veith, Vienna University of Technology, Austria
Maryna Vladimirova, V.N.Karazin Kharkiv National University, Ukraine
Paul Warren, British Telecom, United Kingdom
Irina Zaretskaya, V.N.Karazin Kharkiv National University, Ukraine

Additional Reviewers

Maria Del Carmen Calatrava Moreno, Vienna University of Technology, Austria

Table of Contents

Preface	3
Organization	5
Table of Contents.....	7
PART I: MAIN ICTERI CONFERENCE	10
I.I Invited Talks	11
Formal Foundations for RDF Evolution and Repair.....	12
Abstraction and Verification of Properties of a Real-Time Java	13
I.II Knowledge Based and Decision Support Systems	14
An Implementation of Agent-Based Ontology Alignment	15
I.III ICT in Research, Industrial Deployment, and Knowledge Transfer	16
Towards the Notion of an Abstract Quantum Automaton	17
Checking Inconsistencies in UML Design	33
Automatic Tests and Practical Tasks Generation in Distance Learning Systems.....	44
Satisfiability Problem in Composition-Nominative Logics of Quantifier-Equational Level.....	56
Efficient Algorithm for Reachability Checking in Modeling	71
The Information System as a Tool to Manage R&D at the National Academy of Pedagogical Sciences of Ukraine.....	82
Maintainability Metrics of UML Design	96
Combinatorial Strand Algebra in Insertion Modeling System.....	102
An Approach to Parallelizing Fortran Programs using Rewriting Rules Technique	112

Conceptualization of University Structure as a Complex Mechanism Serving Educational Interests.....	121
On the Problem of Multi-Channel Communication.....	128
KSU Feedback Service as a Tool for Getting Feedback in Educational Institutions. Perspectives of Use in Government Organizations and Commercial Companies ...	134
Issues of Model-Based Distributed Data Processing: Higher Education Resources Evaluation Case Study.....	147
Tested Approach for Variability Management Enhancing in Software Product Line	155
I.IV Methodological and Didactical Aspects of Teaching ICT and Using ICT in Education.....	163
Motivating Students and Improving Quality of Learning Using Peer-Reviews	164
Approach to E-Learning Fundamental Aspects of Software Engineering	176
Choosing the First Educational Programming Language	188
Creation of Multimedia Guides to the History of Music as a Means to form Professional Competence of Future Music Teachers.....	199
An Experience of the Creation and Approbation of the Learning Course “NIT and TFE” for Future Teachers.....	207
Scientific and Educational Project “IT-OSVITA” as a Part of the Training System of Specialists for the Needs of IT Industry of Ukraine	215
Teaching Conceptual Modeling in ER: Chen Worlds	222
Training of Future Primary School Teachers for Application of ICT at Language Lessons	228
The Usage of Educational Portal for Distance Learning	236
I.V Advances in Knowledge-Based and Information Systems.....	243
OntoElect Approach for Iterative Ontology Refinement: a Case Study with ICTERI Scope Ontology	244
An Advanced Active Data Dictionary Based Framework for Flexible Corporate Systems.....	245

PART II: ICTERI WORKSHOPS.....	246
II.I First International Workshop on Dynamics and Evolution in Intelligent Systems (DEIS)	247
Foreword.....	248
Quality of an Ontology as a Dynamic Optimisation Problem	249
II.II First International Workshop on Integration of Information Technologies in Economic Research (ITER).....	257
Foreword.....	258
Direct and Indirect Impact Analysis of Ukrainian Industries on Gross Output and Labor Market in Leontief Model	259
Econometric Analysis of Factors which Determine a Choice of Entrants	267
II.III Workshop on Algebraic, Logical, and Algorithmic Methods of System Modeling, Specification and Verification (SMSV).....	273
Foreword.....	274
A Case Study in Combining Formal Verification and Model-Driven Engineering..	275
Intelligent Testing in Kyiv: Analytical and Deductive Paradigms and their Implementation.....	290
Semantics-based Logics over Hierarchical Nominative Data.....	296
On Existence of Global-in-Time Trajectories of Non-deterministic Markovian Systems.....	312
Verification of Systems: Deadlock Analysis Based on Petri Nets.....	321
Decomposition and Isomorphism of Logical Systems	344
AUTHOR INDEX.....	352

PART I: Main ICTERI Conference

I.I Invited Talks

Formal Foundations for RDF Evolution and Repair

Grigoris Antoniou

University of Huddersfield, United Kingdom
antoniou@ics.forth.gr

Abstract: There are ongoing efforts to provide declarative formalisms of integrity constraints over RDF data. In this context, addressing the evolution of RDF knowledge bases while respecting associated constraints is a challenging issue, yet to receive a full formal treatment. This problem has become more important with the emergence of Linked Open Data, which is based on RDF. Linked Open Data is becoming a key enabler as it is rapidly being taken up by governments and organizations to make their information available and usable, and is the basis of significant economic activity around an emerging "data economy".

In this talk we describe a theoretical framework for dealing with both schema and data change requests, based on the notion a rational change operator as one that satisfies the belief revision principles of Success, Validity and Minimal Change. The semantics of such an operator is subject to customization, by tuning the properties that a rational change should adhere to. We prove some interesting theoretical results, as well as algorithmic solutions. We then show how the problem of ontology repair can be addressed with similar techniques. And we conclude with a discussion of challenges that need to be addressed to further advance the state of the art.

Keywords. RDF knowledge evolution, integrity constraint, Linked Open Data, data economy, schema and data change, belief revision, ontology repair.

Key Terms. LinkedData, KnowledgeEvolution, KnowledgeRepresentation, KnowledgeManagementProcess.

Abstraction and Verification of Properties of a Real-Time Java

Martin Strecker

IRIT-ACADIE, Paul Sabatier University, Toulouse, France

martin.strecker@irit.fr

<http://www.irit.fr/~Martin.Strecker/>

Abstract: The talk will give an overview of ongoing work on verification of concurrent real-time Java programs. Uncontrolled access to shared objects by concurrently executing threads may lead to data incoherencies. We propose to annotate program sections of Java threads with temporal information indicating their activation times. We map this information to Timed Automata (TA) and can then verify by model checking whether the considered program may display resource access conflicts. In our talk, we will describe this approach and present first steps towards a formal verification of the soundness of this abstraction, by modeling the semantics of the formalisms (Java and TA) in a proof assistant.

Keywords. Formal verification, concurrent Java program, data incoherence, Timed Automata, resource access conflict, semantics, proof assistant.

Key Terms. FormalMethod, VerificationProcess, MathematicalModel, Methodology.

I.II Knowledge Based and Decision Support Systems

Papers invited from UNISCON 2012

An Implementation of Agent-Based Ontology Alignment¹

Maxim Davidovsky¹, Vadim Ermolayev², and Vyacheslav Tolok³,

¹Zaporozhye National University, Center of Information Technologies,
Zhukovskogo st. 66, 69600 Zaporozhye, Ukraine

²Zaporozhye National University, Department of Information Technologies,
Zhukovskogo st. 66, 69600 Zaporozhye, Ukraine

³Zaporozhye National University, Department of Mathematical Modelling,
Zhukovskogo st. 66, 69600 Zaporozhye, Ukraine
m.davidovsky@gmail.com, vadim@ermolayev.com,
vyacheslav-tolok@yandex.ru

Abstract. Various knowledge-based information systems contain distinct knowledge representations reflecting different domains of interest and different viewpoints across domains of discourse. For efficient use of knowledge-based systems it is necessary to know semantic relations or alignment between different knowledge representations. One of the promising approaches is the use of intelligent software agents where agents communicate in order to align respective knowledge representations. The paper presents an approach for ontology alignment based on implementation of meaning negotiation between intelligent agents. In the approach, negotiation leads in iterative way. On each step agents compare ontological contexts and use propositional substitutions in order to reduce semantic distance between the contexts. The focus of the paper is the implementation of agents' negotiation strategy.

Keywords. Ontology, ontology alignment, intelligent agent, meaning negotiation, implementation.

Key Terms. KnowledgeRepresentation, MultiAgentSystem, Collaboration

¹ This UNISCON 2012 paper has been invited for a presentation at ICTERI 2012. The paper will appear in full in the post-proceedings of UNISCON 2012.

I.III ICT in Research, Industrial Deployment, and Knowledge Transfer

Towards the Notion of an Abstract Quantum Automaton

Mizal Alobaidi¹, Andriy Batyiev², and Grygoriy Zholtkevych²

¹ Tikrit University, Faculty of Computer Sciences and Mathematics,
P.O. Box--42, Tikrit, Iraq
mizalobaidi@yahoo.com

² V.N. Karazin Kharkiv National University, School of Mathematics
and Mechanics, 4, Svobody Sqr., Kharkiv, 61077, Ukraine
generatorglukoff@gmail.com, zholtkevych@univer.kharkov.ua

Abstract. The main goal of this paper is to give a rigorous mathematical description of systems for processing quantum information. To do it authors consider abstract state machines as models of classical computational systems. This class of machines is refined by introducing constrains on a state structure, namely, it is assumed that state of computational process has two components: a control unit state and a memory state. Then authors modify the class of models by substituting the deterministic evolutionary mechanism for a stochastic evolutionary mechanism. This approach can be generalized to the quantum case: one can replace transformations of a classical memory with quantum operations on a quantum memory. Hence the authors come to the need to construct a mathematical model of an operation on the quantum memory. It leads them to the notion of an abstract quantum automaton. Further the authors demonstrate that a quantum teleportation process is described as evolutionary process for some abstract quantum automaton.

Key words: computational process, computational model, abstract state machine, finite-level quantum system, qubit, Kraus' family, quantum operation, abstract quantum automaton, quantum teleportation

Key terms: MathematicalModelling, MathematicalModel, FormalMethod

1 Introduction

The idea to build a device capable "to compute all that can be computed" had emerged a long time. One can remember Blaise Pascal's Arithmetic Machine, Gottfried Wilhelm Leibniz's Stepped Reckoner, Charles Babbage's Difference Engine and his Analytical Engine [17]. But only in the thirties of last century, Alonzo Church [3], Alan Mathison Turing [15], and Emil Leon Post [14] built mathematical models of the computational processes. Although these models have different shapes each of them describes inherently the same class of processes. The equivalence of the Turing's model and the Church's model, for example, was proved by A.M. Turing in 1937 [16]. In the late forties, hardware implementations of a universal computational system were developed and began to be used. They are known now as computers.

The practice of using computers to solve real problems showed that besides answering the question "Can the problem be solved using computer?", an answer to the question "Do we have enough computational resources to solve the problem?" is important too. Searches for methods to evaluate computational resources for computer-assisted problem solving led to the special scientific area which is called theory of computational complexity (the brief historical overview one can see in [6]). Unfortunately, most important computational problems are complex ones. In compliance with the generally accepted propositions of theoretical computing science the application field of classical computers, i.e. hardware implementations of the universal Turing machine concept, is physically challenged by problems which have polynomial computational complexity.

However, modern science, technique, and technology are in need of methods to solve problems whose complexity is higher than polynomial. This situation stimulates research of non-classical approaches to computing, and quantum computing is one of these.

The idea to use quantum systems as computing devices appeared in the early eighties of the twentieth century. The idea's authors considered it as a way to overcome computational complexity. In the context Yuri Ivanovitch Manin's monograph¹ [11] and Richard Phillips Feynman's paper [4] should be noted. Considering the possibility of using quantum machines for solving complex problems of simulation Yu.I. Manin wrote (cited by [12]): "we need a mathematical theory of quantum automata. Such a theory would provide us with mathematical models of deterministic processes with quite unusual properties. One reason for this is that the quantum state space has far greater capacity than the classical one: for a classical system with N states, its quantum version allowing superposition (entanglement) accommodates e^N states". In [12], Yu.I. Manin also sets requirements to the mathematical theory of quantum automata: "The first difficulty we must overcome is the choice of the correct balance between the mathematical and the physical principles. The quantum automaton has to be an abstract one: its mathematical model must appeal only to the general principles of quantum physics, without prescribing a physical implementation. Then the model of evolution is the unitary rotation in a finite dimensional Hilbert space, and the decomposition of the system into its virtual parts corresponds to the tensor product decomposition of the state space ("quantum entanglement"). Somewhere in this picture we must accommodate interaction, which is described by density matrices and probabilities".

R.P. Feynman had a similar opinion [4, 5].

This paper is an attempt to construct a mathematical model of quantum automata that fulfils requirements formulated by Yu.I. Manin.

2 Classical Computational Model

In this section a mathematical model of a classical computational system is considered. The approach was proposed by A.N. Kolmogorov and V.A. Uspensky

¹ The monograph's introduction was translated into English [12]

[10]. Theory of abstract state machine (ASM) is the further development of the approach [7, 8].

2.1 Preliminary definitions

Definition 1. Let \mathbf{A} denotes an algorithm. It is determined by

- a set $C(\mathbf{A})$ of states;
- a subset $I(\mathbf{A})$ of $C(\mathbf{A})$ which elements are called initial states;
- a subset $T(\mathbf{A})$ of $C(\mathbf{A})$ which elements are called terminal states;
- a map $\tau_{\mathbf{A}} : C(\mathbf{A}) \rightarrow C(\mathbf{A})$ which defines one step of the computational process;
- and the next condition

$$I(\mathbf{A}) \cap T(\mathbf{A}) = \emptyset. \quad (1)$$

Note that elements of the set $C(\mathbf{A})$ correspond to complete state descriptions of the computational process which is defined by the algorithm \mathbf{A} .

Definition 2. Let \mathbf{A} be an algorithm then a partial map $C : \mathbf{N} \rightarrow C(\mathbf{A})$ ² is called a **run** of the algorithm if it satisfies the following conditions

- $C(0) \in I(\mathbf{A})$;
- if $C(t) \neq \emptyset$ for some $t \in \mathbf{N}$ then $C(t') \neq \emptyset$ for all $t' \in \mathbf{N}$ such that $t' < t$;
- if $C(t+1) \neq \emptyset$ for some $t \in \mathbf{N}$ then $C(t+1) = \tau_{\mathbf{A}}(C(t))$;
- if $\emptyset \neq C(t) \in T(\mathbf{A})$ then $C(t+1) = \emptyset$.

From this definition it follows immediately that the domain of an arbitrary run is the set \mathbf{N} or some set $0..T = \{t \in \mathbf{N} \mid t \leq T\}$, where T is a non-negative integer.

In the first case the algorithm **diverges** on the initial state $C(0)$ (this is denoted by $\mathbf{A}(C(0)) \uparrow$).

In the second case the algorithm **converges** on the initial state $C(0)$ to $C(T)$ (this is denoted by $\mathbf{A}(C(0)) \downarrow C(T)$).

2.2 Abstract state machines with stochastic behaviour

Let's refine the Definition 1 and Definition 2 for such algorithms that have sets of states with some special structure.

Let's start refining with the following auxiliary definitions.

² For two sets A and B by $f : A \rightarrow B$ a partial map from A into B is denoted. For $a \in A$ by $f(a) \neq \emptyset$ the clause " $f(a)$ is defined" is denoted.

Definition 3. Let N and A be finite sets of nodes and arcs respectively, dom and codom be maps that associate with arcs their initial and terminal nodes respectively, then the tuple $(N, A, \text{dom}, \text{codom})$ is called a directed multigraph.

Definition 4. Let $G = (N, A, \text{dom}, \text{codom})$ be a directed multigraph, then an alternating sequence $\alpha = n_1, a_1, \dots, a_k, n_{k+1}$ of nodes and arcs, beginning and ending with a node, is called a walk if for all $s = 1, \dots, k$ the next condition holds: $\text{dom}(a_s) = n_s$ and $\text{codom}(a_s) = n_{s+1}$.

In this case we shall use the notation: $n_1 \xrightarrow{a_1} \dots \xrightarrow{a_k} n_{k+1}$.

Definition 5. Let $\alpha = n_1 \xrightarrow{a_1} \dots \xrightarrow{a_k} n_{k+1}$ be a walk in the directed multigraph $G = (N, A, \text{dom}, \text{codom})$ and n be its node, then we shall say that

- n is the initial node of α (it is denoted by $\text{dom}(\alpha) = n$) if $n = n_1$;
- n is the terminal node of α (it is denoted by $\text{codom}(\alpha) = n$) if $n = n_{k+1}$;
- α traverses n if for some $s \in \{1, \dots, k+1\}$ the equality $n = n_s$ holds.

Definition 6. Let $(N, A, \text{dom}, \text{codom}, n_0, F)$ be a tuple such that the tuple $(N, A, \text{dom}, \text{codom})$ is a directed multigraph, n_0 is a fixed node (it is called the initial node), F is a fixed subset of nodes (its elements are called terminal nodes). The tuple is called a control graph if the next conditions hold:

- $n_0 \notin F$;
- for each $n \in F$ there is no arc with initial node equals n ;
- for each node n there is a walk such that its initial node equals n_0 , its terminal node belongs to F , and it traverses n .

Note that for the control graph $(N, A, \text{dom}, \text{codom}, n_0, F)$ and each $n \in N \setminus F$ the set $\text{Out}(n) = \{a \in A \mid \text{dom}(a) = n\}$ is not empty.

Let's assume that for an arbitrary algorithm \mathbf{A} the set of states has the next structure $\mathbf{C}(\mathbf{A}) = N(\mathbf{A}) \times S(\mathbf{A})$ where $N(\mathbf{A})$ is the nodes set of some control graph $G(\mathbf{A})$ and $S(\mathbf{A})$ is some set of memory snapshots.

In this case suppose that the set of initial states is the next set $I(\mathbf{A}) = \{(n_0, S) \mid S \in S(\mathbf{A})\}$ and the set of terminal states is the following set $T(\mathbf{A}) = \{(n, S) \mid n \in F \ \& \ S \in S(\mathbf{A})\}$.

This supposition leads us to the next representation of the map $\tau_{\mathbf{A}}$:

$$\tau_{\mathbf{A}}(n, S) = (\sigma_{\mathbf{A}}(n, S), \gamma_{\mathbf{A}}(n, S)), \text{ where } \sigma_{\mathbf{A}} : N(\mathbf{A}) \times S(\mathbf{A}) \rightarrow N(\mathbf{A}) \quad (2)$$

$$\text{and } \gamma_{\mathbf{A}} : N(\mathbf{A}) \times S(\mathbf{A}) \rightarrow S(\mathbf{A})$$

Suppose now that the map $\sigma_{\mathbf{A}}$ has property of locality. It means that for each $n \in N(\mathbf{A}) \setminus F$ there exists a map $h_n : \mathcal{S}(\mathbf{A}) \rightarrow \text{Out}(n)$ and for each $a \in \mathcal{A}(\mathbf{A})$ ³ there exists a map $g_a : \mathcal{S}(\mathbf{A}) \rightarrow \mathcal{S}(\mathbf{A})$ such that the following equalities are true:

$$\sigma_{\mathbf{A}}(n, S) = \text{codom}(h_n(S)); \quad (3)$$

$$\gamma_{\mathbf{A}}(n, S) = g_{h_n(S)}(S). \quad (4)$$

From (2), (3), and (4) it follows

$$\tau_{\mathbf{A}}(n, S) = (\text{codom}(h_n(S)), g_{h_n(S)}(S)). \quad (5)$$

Therefore, we can consider the computational process which is determined by the algorithm \mathbf{A} as a sequence of steps. Each step begins when the current state is described by some control graph node n and a memory snapshot S . Then the map h_n chooses the arc a outgoing from the node n depending on the snapshot S . Finally, using the selected arc and the memory snapshot the new control graph node and the new memory snapshot are determined in compliance with (5).

Let's modify the computational model by rejecting the assumption about determinacy for the choosing process. Definition 2.5 describes this modification formally.

Definition 7. Let $\mathbf{G} = (N, \mathcal{A}, \text{dom}, \text{codom}, n_0, F)$ be a control graph, \mathcal{S} be some set of memory snapshots, $\mathbf{P} = \{\text{Pr}(\cdot | S, n) \mid n \in N \setminus F, S \in \mathcal{S}\}$ be a family of probability distributions on \mathcal{A} , and $\mathbf{T} = \{g_a \mid a \in \mathcal{A}\}$ be a family of maps from \mathcal{S} into itself then the tuple $(\mathbf{G}, \mathcal{S}, \mathbf{P}, \mathbf{T})$ is called an abstract state machine with stochastic behaviour if the following condition holds

$$\text{for all } n \in N, S \in \mathcal{S}, a \in \mathcal{A} \text{ if } a \notin \text{Out}(n) \text{ then } \text{Pr}(a | S, n) = 0. \quad (6)$$

Dynamics of such machines is determined by the next definition.

Definition 8. Let $(\mathbf{G}, \mathcal{S}, \mathbf{P}, \mathbf{T})$ be an abstract state machine with stochastic behaviour, $\mathbf{G} = (N, \mathcal{A}, \text{dom}, \text{codom}, n_0, F)$ be its control graph then a partial map $C : \mathbf{N} \rightarrow N \times \mathcal{S}$ is called a run of the machine if it satisfies the following conditions

- $C(0) = (n_0, S)$, where $S \in \mathcal{S}$;
- if $C(t) \neq \emptyset$ for some $t \in \mathbf{N}$ then $C(t') \neq \emptyset$ for all $t' \in \mathbf{N}$ such that $t' < t$;
- if $\emptyset \neq C(t+1) = (n', S')$ for some $t \in \mathbf{N}$ and $C(t) = (n, S)$ then there exists $a \in \text{Out}(n)$ such that $\text{Pr}(a | n, S) > 0$, $n' = \text{codom}(a)$, and $S' = g_a(S)$;
- if $\emptyset \neq C(t) = (n, S)$ for $n \in F$ and $S \in \mathcal{S}$ then $C(t+1) = \emptyset$.

Below such machines will be generalised for the quantum case.

³ By $\mathcal{A}(\mathbf{A})$ is denoted the set of arcs for the control graph $\mathbf{G}(\mathbf{A})$.

3 Mathematical Model of Finite-Level Quantum Systems

In the section the model of quantum systems with finite quantity of levels (finite-level quantum systems) is described. It is based on the approaches set forth in the works [9, 13].

3.1 Postulates of finite-level quantum systems

The postulates of finite-level quantum systems fix basic notions which are used to construct mathematical models for the systems.

Postulate 1: an n -dimensional Hilbert space H_n is associated to any quantum physical system with n levels. This space is known as the state space of the system. The system is completely described by its pure state, which is a one-dimensional subspace of the state space. This subspace is uniquely represented by the ortho-projector $|\psi\rangle\langle\psi|$ on a vector $|\psi\rangle$ which generates the subspace.

In contrast to pure states mixed states are used to describe quantum systems whose state is not completely known.

Rather more detailed suppose we know that a quantum system is in one of a number of states $\{|\psi_k\rangle\langle\psi_k| : k=1, \dots, m\}$ with respective probabilities $\{p_k : k=1, \dots, m\}$. We shall call $\{p_k, |\psi_k\rangle\langle\psi_k| : k=1, \dots, m\}$ an ensemble of pure states. The density operator for the system is defined by the equation

$$\rho = \sum_{k=1}^m p_k |\psi_k\rangle\langle\psi_k|.$$

We identify mixed states with density operators ⁴. Evidently, that each density operator is a non-negative defined operator which trace is equal to unit. It is known that the inverse statement is true: a non-negative defined operator, which trace is equal to unit, is a density operator [9].

Of course, a one-dimensional ortho-projector is a density operator. The set of all density operators is convex and its subset of one-dimensional ortho-projectors is the subset of its extreme points [9]. This allows to consider pure states as indecomposable states.

Postulate 2: the state space of a composite physical system is the tensor product of the state spaces of the component physical systems. Moreover, if we have systems indexed by $k=1, \dots, m$, and the state of the system with number k is described by the density operator ρ_k , then the joint state of the total system before any interactions is $\rho_1 \otimes \dots \otimes \rho_m$.

Postulate 3: the evolution of a closed quantum system is described by a unitary transformation. That is, the state $|\psi\rangle\langle\psi|$ of the system at time t_1 is related to the

⁴ The set of all density operators on the space H_n is denoted by \mathbf{S}_n

state $|\psi'\rangle\langle\psi'|$ of the system at time t_2 by a unitary operator U which depends only on the times t_1 and t_2 , $|\psi'\rangle\langle\psi'| = U|\psi\rangle\langle\psi|U^*$.

If we have an ensemble of pure states of the system which is described by the density operator ρ at time t_1 then the density operator ρ' of the system at time t_2 can be calculated by the formula $\rho' = U\rho U^*$.

Postulate 4: quantum measurements are described by an indexed finite family $\mathbf{K} = \{K(x) : x \in X\}$ of Kraus' operators, where X is a finite set. These are operators acting on the state space of the system being measured. The index x refers to the measurement outcome that may occur in the experiment. If the state of the quantum system is described by the density operator ρ immediately before the measurement then the probability that result x occurs is given by the following formula⁵

$$\Pr(x|\rho) = \text{Tr}\left(\rho K(x)^* K(x)\right) \quad (7)$$

and the state of the system immediately after the measurement is described by the density operator

$$\text{Eff}[\rho|x] = \frac{K(x)\rho K(x)^*}{\text{Tr}\left(\rho K(x)^* K(x)\right)} \quad (8)$$

Any Kraus' family $\mathbf{K} = \{K(x) : x \in X\}$ satisfies the **completeness condition**

$$\sum_{x \in X} K(x)^* K(x) = \mathbf{1} \quad (9)$$

which ensures correctness of the definitions given by formulas (7) and (8).

3.2 Measurements and isometric operators. Quantum operations

Postulate 3 and Postulate 4 describe two different ways of changing a system state. It looks non-naturally. Hence, we can set the problem: find a unified description for evolutions and measurements of a finite-level quantum system.

To solve the problem let's introduce for a state space \mathbf{H}_n of an n -level quantum system and a finite set X operators $J(x) : \mathbf{H}_n \rightarrow \mathbf{H}_n \otimes l^2(X)$ by the formula

$$J(x)|\psi\rangle = |\psi\rangle \otimes |x\rangle \quad (10)$$

where $|x\rangle \in l^2(X)$ such that $|x\rangle(\cdot) = \delta(x, \cdot)$.

⁵ By $\text{Tr}(\cdot)$ the usual operator trace is denoted

Properties of operators from the family $\{J(x) : x \in X\}$ are established by the next proposition, which is proved by the direct calculation.

Proposition 1. Let \mathbf{H}_n be a state space of an n -level quantum system and X be a finite set, then the operators family $\{J(x) : x \in X\}$ defined by formula (10) satisfies the next identities

$$J(x)^* \sum_{x' \in X} (|\psi(x')\rangle \otimes |x'\rangle) = |\psi(x)\rangle \quad (11)$$

$$J(x')^* J(x'') = \delta(x', x'') \cdot \mathbf{1} \quad (12)$$

$$J(x') J(x'')^* = \mathbf{1} \otimes |x'\rangle \langle x''| \quad (13)$$

Now using a Kraus' family $\mathbf{K} = \{K(x) : x \in X\}$ for some measurement let's define an operator $W_{\mathbf{K}} : \mathbf{H}_n \rightarrow \mathbf{H}_n \otimes l^2(X)$ by the formula

$$W_{\mathbf{K}} |\psi\rangle = \sum_{x' \in X} K(x') |\psi\rangle \otimes |x'\rangle \quad (14)$$

Proposition 2. Let $\mathbf{K} = \{K(x) : x \in X\}$ be a Kraus' family, and $W_{\mathbf{K}}$ be the operator that is built by the formula (14), then $W_{\mathbf{K}}$ is an isometric operator and the next identities hold for all $x \in X$:

$$K(x) = J(x)^* W_{\mathbf{K}} \quad (15)$$

Proof. Let $|0\rangle, \dots, |n-1\rangle$ be some orthonormal basis in \mathbf{H}_n , then for any $k = 0, \dots, n-1$ from (14) we have

$$W_{\mathbf{K}} |k\rangle = \sum_{x' \in X} K(x') |k\rangle \otimes |x'\rangle$$

Hence,

$$W_{\mathbf{K}} = \sum_{k=0}^{n-1} \sum_{x' \in X} (K(x') |k\rangle \otimes |x'\rangle) \langle k|$$

and

$$W_{\mathbf{K}}^* = \sum_{k=0}^{n-1} \sum_{x' \in X} |k\rangle \langle k| K(x')^* \otimes \langle x'|$$

Therefore,

$$\begin{aligned}
W_{\mathbf{K}}^* W_{\mathbf{K}} &= \left(\sum_{l=0}^{n-1} \sum |l\rangle \langle l| K(x'')^* \otimes \langle x''| \right) \cdot \left(\sum_{k=0}^{n-1} \sum_{x' \in X} (K(x')|k\rangle \otimes |x'\rangle) \langle k| \right) = \\
& \sum_{k,l=0}^{n-1} \sum_{x',x'' \in X} |l\rangle \langle l| K(x'')^* K(x')|k\rangle \langle x''|x'\rangle \langle k| = \sum_{k=0}^{n-1} \sum_{x' \in X} K(x'')^* K(x')|k\rangle \langle k| = \\
& \sum_{x' \in X} K(x')^* K(x').
\end{aligned}$$

Using the completeness condition one can obtain that $W_{\mathbf{K}}^* W_{\mathbf{K}} = \mathbf{1}$.

The last equation ensures that $W_{\mathbf{K}}$ is an isometric operator.

Equation (15) is proved by the direct calculation:

$$J(x)^* W_{\mathbf{K}} |\psi\rangle = J(x)^* \sum_{x' \in X} (K(x')|\psi\rangle \otimes |x'\rangle) = K(x)|\psi\rangle.$$

Proof is complete.

Using Proposition 2 one can rewrite formulae (7) and (8) in the following way:

$$\Pr(x|\rho) = \text{Tr}(\rho W_{\mathbf{K}}^* W_{\mathbf{K}} (\mathbf{1} \otimes |x\rangle \langle x|)). \quad (16)$$

$$\text{Eff}[\rho|x] = \frac{J(x)^* W_{\mathbf{K}} \rho W_{\mathbf{K}}^* J(x)}{\text{Tr}(\rho W_{\mathbf{K}}^* (\mathbf{1} \otimes |x\rangle \langle x|) W_{\mathbf{K}})}. \quad (17)$$

Now we claim that this construction can be inverted.

Really, let \mathbf{H}_n be a state space for an n -level quantum system, X be a finite set of outcomes, and $W : \mathbf{H}_n \rightarrow \mathbf{H}_n \otimes l^2(X)$ be an isometric operator.

Let's define a family $\mathbf{K} = \{K(x) : x \in X\}$ of operators on the space \mathbf{H}_n by the formula

$$K(x) = J(x)^* W. \quad (18)$$

Proposition 3. Let \mathbf{H}_n be a state space for an n -level quantum system, X be a finite set of outcomes, $W : \mathbf{H}_n \rightarrow \mathbf{H}_n \otimes l^2(X)$ be an isometric operator, and $\mathbf{K} = \{K(x) : x \in X\}$ be the family of operators which is defined by formula (16); then

1. \mathbf{K} satisfies the completeness condition and, therefore, it is a Kraus' family;
2. $W_{\mathbf{K}} = W$.

Proof. To prove the completeness condition let's calculate the left side of (9) using (13) and the isometry property

$$\sum_{x \in X} K(x)^* K(x) = \sum_{x \in X} W^* J(x) J(x)^* W = \sum_{x \in X} W^* (\mathbf{1} \otimes |x\rangle \langle x|) W = W^* W = \mathbf{1}.$$

To prove the second statement let's calculate using (13)

$$\begin{aligned} W_K |\psi\rangle &= \sum_{x \in X} (K(x)|\psi\rangle \otimes |x\rangle) = \sum_{x \in X} J(x)K(x)|\psi\rangle = \\ \sum_{x \in X} J(x)(J(x)^*W)|\psi\rangle &= \sum_{x \in X} (J(x)J^*(x)W)|\psi\rangle = \sum_{x \in X} (\mathbf{1} \otimes |x\rangle\langle x|)W|\psi\rangle = W|\psi\rangle. \end{aligned}$$

Proof is complete.

Proposition 2 and 3, formulae (16) and (17) substantiate replacing the Kraus' families by the corresponding isometric operators under studying the interaction of quantum systems with classical systems. This replacing leads us to unification of Postulate 3 and Postulate 4. To stress such unification we will say that an isomeric operator describes the quantum operation by formulae (16) and (17).

Definition 9. Let \mathbf{H}_n be a state space of an n -level quantum system, X be a finite set of outcomes, then isometric operators $W_1, W_2 : \mathbf{H}_n \rightarrow \mathbf{H}_n \otimes l^2(X)$ are called equivalent if for all $x \in X$ and for any density operator ρ the following equalities are true

$$\text{Tr}(\rho W_1^* (\mathbf{1} \otimes |x\rangle\langle x|) W_1) = \text{Tr}(\rho W_2^* (\mathbf{1} \otimes |x\rangle\langle x|) W_2), \quad (19)$$

$$J(x)^* W_1 \rho W_1^* J(x) = J(x)^* W_2 \rho W_2^* J(x). \quad (20)$$

Classes of this equivalence will be called **quantum operations** with a set of outcomes X .

Easy to see that isometric operators $W_1, W_2 : \mathbf{H}_n \rightarrow \mathbf{H}_n \otimes l^2(X)$ describe the same quantum operation if $J(x)^* W_2 = e^{i\theta(x)} J(x)^* W_1$ for any $\theta : X \rightarrow [0, 2\pi)$.

We claim that the inverse statement is true too.

Theorem 1. Let \mathbf{H}_n be a state space of an n -level quantum system, X be a finite set of outcomes, $W_1, W_2 : \mathbf{H}_n \rightarrow \mathbf{H}_n \otimes l^2(X)$ be equivalent isometric operators then $J(x)^* W_2 = e^{i\theta(x)} J(x)^* W_1$ for some $\theta : X \rightarrow [0, 2\pi)$.

Proof. It is evident, that each isometric operator $W_s : \mathbf{H}_n \rightarrow \mathbf{H}_n \otimes l^2(X)$, where $s = 1, 2$, can be represented by the formula

$$W_s = \sum_{x \in X} \sum_{k=0}^{n-1} (|\omega_k^{(s)}(x)\rangle \otimes |x\rangle) \langle k|, \quad (21)$$

where $\{|0\rangle, \dots, |n-1\rangle\}$ is an orthonormal basis in \mathbf{H}_n and $|\omega_k^{(s)}(x)\rangle = J(x)^* W_s |k\rangle$ for $k = 0, \dots, n-1$ and $x \in X$.

Using representation (21) we calculate $\text{Pr}(x ||k\rangle\langle k|)$ for W_s where $s = 1, 2$.

$$\begin{aligned}
\Pr(x | |k\rangle\langle k|) &= \text{Tr}\left(|k\rangle\langle k| W_s^* (\mathbf{1} \otimes |x\rangle\langle x|) W_s\right) = \langle k | W_s^* (\mathbf{1} \otimes |x\rangle\langle x|) W_s | k \rangle = \\
&= \langle k | W_s^* (\mathbf{1} \otimes |x\rangle\langle x|) \sum_{x' \in X} \sum_{l=0}^{n-1} \left(|\omega_l^{(s)}(x')\rangle \otimes |x'\rangle \right) \langle l | k \rangle = \langle k | W_s^* \left(|\omega_k^{(s)}(x)\rangle \otimes |x\rangle \right) = \\
&= \langle k | \sum_{x' \in X} \sum_{l=0}^{n-1} |l\rangle \left(\langle \omega_l^{(s)}(x') | \otimes \langle x' | \right) \left(|\omega_k^{(s)}(x)\rangle \otimes |x\rangle \right) = \|\omega_k^{(s)}(x)\|^2.
\end{aligned}$$

Using this and identity (19) one can derive that for all $k = 0, \dots, n-1$ and $x \in X$ the next equality holds

$$\|\omega_k^{(1)}(x)\|^2 = \|\omega_k^{(2)}(x)\|^2 \quad (22)$$

Let $I_s(x) = \{k \mid 0 \leq k < n, \|\omega_k^{(s)}(x)\| \neq 0\}$ for each $x \in X$ and $s = 0, 1$. From (22) it follows that $I_1(x) = I_2(x)$, hence, we can denote this set by $I(x)$.

From (20) one can derive that for all $x \in X$ and $k \in I(x)$

$$|\omega_k^{(2)}(x)\rangle \langle \omega_k^{(2)}(x)| = |\omega_k^{(1)}(x)\rangle \langle \omega_k^{(1)}(x)|. \quad (23)$$

The next equality is obtained by multiplying equality (23) from left by $\langle \omega_k^{(1)}(x)|$ and from right by $|\omega_k^{(1)}(x)\rangle$ and using equality (22):

$$\left| \langle \omega_k^{(2)}(x) | \omega_k^{(1)}(x) \rangle \right|^2 = \|\omega_k^{(2)}(x)\|^2 \cdot \|\omega_k^{(1)}(x)\|^2. \quad (24)$$

From the (24) and (22) it follows that for all $x \in X$ and $k \in I(x)$

$$|\omega_k^{(2)}(x)\rangle = e^{i\theta(k,x)} |\omega_k^{(1)}(x)\rangle, \text{ where } 0 \leq \theta(k,x) < 2\pi. \quad (25)$$

Further, from (20) it follows that for all $x \in X$ and $k, l \in I(x)$ the next equality is true:

$$|\omega_k^{(2)}(x)\rangle \langle \omega_l^{(2)}(x)| = |\omega_k^{(1)}(x)\rangle \langle \omega_l^{(1)}(x)|.$$

Therefore,

$$e^{i(\theta(k,x) - \theta(l,x))} |\omega_k^{(1)}(x)\rangle \langle \omega_l^{(1)}(x)| = |\omega_k^{(1)}(x)\rangle \langle \omega_l^{(1)}(x)|$$

and $e^{i(\theta(k,x) - \theta(l,x))} = 1$.

In summary, we obtain the next equality for all $x \in X$ and $k \in I(x)$

$$|\omega_k^{(2)}(x)\rangle = e^{i\theta(x)} |\omega_k^{(1)}(x)\rangle. \quad (26)$$

Using (24) for $x \in X$, $k \in I(x)$ and the equality $|\omega_l^{(2)}(x)\rangle = |\omega_l^{(1)}(x)\rangle = 0$ for $l \in \{0, \dots, n-1\} \setminus I(x)$ one can get that equality (26) is true for all $0 \leq k < n$.

Therefore, $J(x)^* W_2 = e^{i\theta(x)} J(x)^* W_1$ for some $\theta: X \rightarrow [0, 2\pi)$.

Corollary 1. Two isometric operators $W_1, W_2: \mathbf{H}_n \rightarrow \mathbf{H}_n \otimes l^2(X)$ define the same quantum operation if and only if for some $\theta: X \rightarrow [0, 2\pi)$ the following equality holds $W_2 = \Theta W_1$, where $\Theta = \mathbf{1} \otimes \sum_{x \in X} e^{i\theta(x)} |x\rangle\langle x|$.

4 Abstract Quantum Automata

Now we describe some class of mathematical models for quantum information processes. This class we call the class of abstract quantum automata.

4.1 The notion of an abstract quantum automaton

Definition 10. Let $\mathbf{H}_m (m > 1)$ be a state space of an m -level quantum system, and let $\mathbf{G} = (N, \mathcal{A}, \text{dom}, \text{codom}, n_0, F)$ be a control graph. Suppose that each non-terminal node n of the graph \mathbf{G} is connected with a quantum operation for which \mathbf{H}_m is the state space, $\text{Out}(n)$ is the outcomes set, and $W_n: \mathbf{H}_m \rightarrow \mathbf{H}_m \otimes l^2(\text{Out}(n))$ is an isometric operator describing the operation. Then the tuple

$$(\mathbf{H}_m, \mathbf{G}, \{W_n \mid n \in N \setminus F\})$$

is called an abstract quantum automaton.

The next definition describes the set of runs for an abstract quantum automaton similarly to Definition 8.

Definition 11. Let $(\mathbf{H}_m, \mathbf{G}, \{W_n \mid n \in N \setminus F\})$ be an abstract quantum automaton where the control graph \mathbf{G} is equal to $(N, \mathcal{A}, \text{dom}, \text{codom}, n_0, F)$. Then a partial map $C: \mathbf{N} \rightarrow N \times \mathbf{S}_m$ is called a run of the automaton if it satisfies the following conditions

- $C(0) = (n_0, \rho)$, where $\rho \in \mathbf{S}_m$;
- if $C(t) \neq \emptyset$ for some $t \in \mathbf{N}$ then $C(t') \neq \emptyset$ for all $t' \in \mathbf{N}$ such that $t' < t$;
- if $\emptyset \neq C(t+1) = (n', \rho')$ for some $t \in \mathbf{N}$ and $C(t) = (n, \rho)$ then there exists $a \in \text{Out}(n)$ such that

$$\Pr(a \mid n, \rho) = \text{Tr}(\rho W_n^* (\mathbf{1} \otimes |a\rangle\langle a|) W_n) > 0, \quad n' = \text{codom}(a),$$

$$\text{and } \rho' = \text{Eff}[\rho|n, a] = \frac{J(a)^* W_n \rho W_n^* J(a)}{\text{Pr}(a|n, \rho)};$$

– if $\emptyset \neq C(t) = (n, \rho)$ for $n \in F$ and $\rho \in \mathbf{S}_m$ then $C(t+1) = \emptyset$.

Now we can consider two important examples.

Let's consider a quantum information process which sets a qubit (2-level quantum system) into the state $|0\rangle\langle 0|$.

Evidently, that this problem can not be solved by any unitary transformation.

We shall specify an abstract quantum automaton that does it. The control graph of the automaton is shown in Fig. 1.

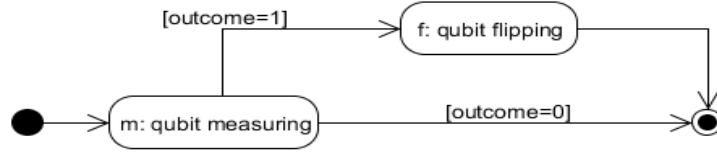


Fig. 1. Qubit cleaner.

As one can see $\text{Out}(m) = \{0,1\}$. Let's define $W_m : \mathbf{H}_2 \rightarrow \mathbf{H}_2 \otimes l^2(\{0,1\})$ by the formula

$$W_m |\psi\rangle = |0\rangle\langle 0|\psi\rangle \otimes |0\rangle + |1\rangle\langle 1|\psi\rangle \otimes |1\rangle.$$

Further, $\text{Out}(f)$ is a singleton hence $W_f : \mathbf{H}_2 \rightarrow \mathbf{H}_2$. Let's define

$$W_f |\psi\rangle = |0\rangle\langle 1|\psi\rangle + |1\rangle\langle 0|\psi\rangle.$$

Easy to see that for an arbitrary initial state of a qubit its state after handling by the automaton is equal to $|0\rangle\langle 0|$.

Therefore, we have built the abstract quantum automaton that specifies the process of cleaning a qubit.

The next example deals with preparing an entangled pair of qubits. We shall specify an abstract quantum automaton that does it.

The control graph of the automaton is shown in Fig. 2.

Let's define $W_h : \mathbf{H}_2 \otimes \mathbf{H}_2 \rightarrow \mathbf{H}_2 \otimes \mathbf{H}_2$ by the formulae

$$W_h (|\psi\rangle \otimes |0\rangle) = |\psi\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle),$$

$$W_h(|\psi\rangle \otimes |1\rangle) = |\psi\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle),$$

and define $W_c : \mathbf{H}_2 \otimes \mathbf{H}_2 \rightarrow \mathbf{H}_2 \otimes \mathbf{H}_2$ by the formulae

$$\begin{aligned} W_c(|\psi\rangle \otimes |0\rangle) &= |\psi\rangle \otimes |0\rangle, \\ W_c(|\psi\rangle \otimes |1\rangle) &= |1\rangle\langle\psi|0\rangle| \otimes |1\rangle + |0\rangle\langle\psi|1\rangle| \otimes |1\rangle. \end{aligned}$$

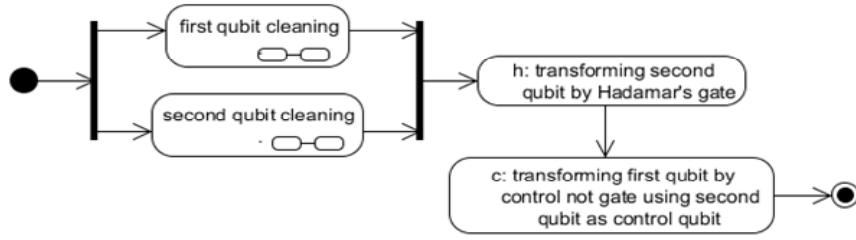


Fig. 2. Preparing an entangled pair of qubits.

Easy to see that for an arbitrary initial state of a qubit pair its state after handling by the automaton is equal to

$$\frac{1}{2}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle)(\langle 0| \otimes \langle 0| + \langle 1| \otimes \langle 1|).$$

These examples demonstrate that modelling of quantum information processes by abstract quantum automata allows to describe processes of initial preparation of a quantum memory for quantum computing devices.

4.2 Quantum teleportation as an abstract quantum automaton

To complete the paper let's consider the quantum teleportation process and let's show that it can be described by an abstract quantum automaton.

Quantum teleportation is a process by which a qubit state can be transmitted exactly from one location to another, without the qubit being transmitted through the intervening space. This phenomenon has been confirmed experimentally [1, 2].

The control graph of the automaton is shown in Fig. 3.

Let's define $W_c : \mathbf{H}_2 \otimes \mathbf{H}_2 \otimes \mathbf{H}_2 \rightarrow \mathbf{H}_2 \otimes \mathbf{H}_2 \otimes \mathbf{H}_2$, where the first qubit is Alice's qubit, the second and the third qubits are the first and the second qubits of the entangled pair respectively, by the formulae

$$\begin{aligned} W_c(|0\rangle \otimes |k\rangle \otimes |\psi\rangle) &= |0\rangle \otimes |k\rangle \otimes |\psi\rangle, \text{ where } k = 0, 1, \\ W_c(|1\rangle \otimes |0\rangle \otimes |\psi\rangle) &= |1\rangle \otimes |1\rangle \otimes |\psi\rangle, \end{aligned}$$

$$W_c(|1\rangle \otimes |1\rangle \otimes |\psi\rangle) = |1\rangle \otimes |0\rangle \otimes |\psi\rangle.$$

Further, $\text{Out}(m) = \{00, 01, 10, 11\}$ and the corresponding isometric operator is defined by formulae

$$W_m(|0\rangle \otimes |0\rangle \otimes |\psi\rangle) = |0\rangle \otimes |0\rangle \otimes |\psi\rangle \otimes |00\rangle,$$

$$W_m(|0\rangle \otimes |1\rangle \otimes |\psi\rangle) = |0\rangle \otimes |1\rangle \otimes |\psi\rangle \otimes |01\rangle,$$

$$W_m(|1\rangle \otimes |0\rangle \otimes |\psi\rangle) = |1\rangle \otimes |0\rangle \otimes |\psi\rangle \otimes |10\rangle,$$

$$W_m(|1\rangle \otimes |1\rangle \otimes |\psi\rangle) = |1\rangle \otimes |1\rangle \otimes |\psi\rangle \otimes |11\rangle.$$

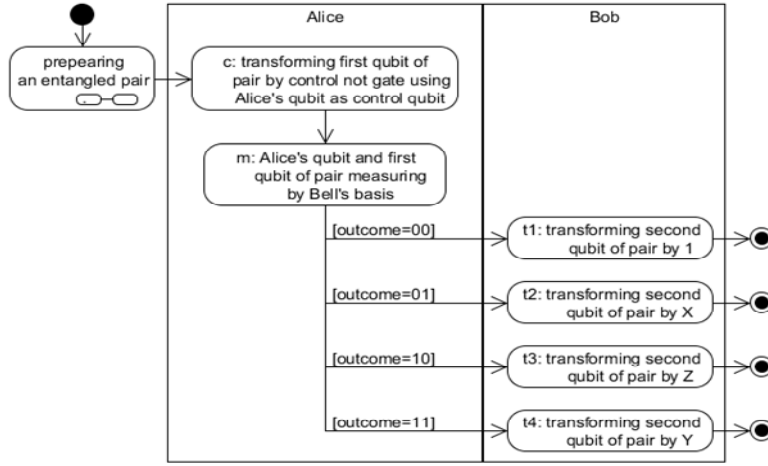


Fig. 3. Teleportation

By direct calculation one can prove that the initial state $|\psi\rangle\langle\psi| \otimes \rho$ for an arbitrary $\rho \in \mathbf{S}_4$ is transformed by the automaton into the state

$$\frac{1}{4}(|0\rangle\langle 0| \otimes |0\rangle\langle 0| + |0\rangle\langle 0| \otimes |1\rangle\langle 1| + |1\rangle\langle 1| \otimes |0\rangle\langle 0| + |1\rangle\langle 1| \otimes |1\rangle\langle 1|) \otimes |\psi\rangle\langle\psi|.$$

Conclusion

Summarising the above we can conclude:

- our attempt to solve the Yu. Manin's problem led us towards the notion of an abstract quantum automaton;
- this notion is based on a computational model known as a machine of A. Kolmogorov and V. Uspensky;

— abstract quantum automata can be used for formal specification of quantum information processes including non-invertible processes like qubit cleaning, entangled pair preparing and quantum teleportation.

The authors know that quantum algorithms can be specified by using abstract quantum automata but corresponding results are not given in the paper because they are cumbersome.

References

1. Bouwmeester, D., Pan, J.-W., Mattle, K., Eible, M., Weinfurter, H., Zeilinger, A.: Experimental quantum teleportation. *Nature*, 390, 575 -- 579 (1997)
2. Boschi, D., Branca, S., De Martini, F., Hardy, L., Popescu, S.: Experimental Realization of Teleporting an Unknown Pure Quantum State via Dual Classical and Einstein--Podolsky--Rosen Channel. *Phys. Rev. Lett.*, 80, 1121 -- 1125 (1998)
3. Church, A.: An unsolvable problem of elementary number theory. *Amer. J. Math.*, 58(2), 345 -- 363 (1936)
4. Feynman, R.P.: Simulating Physics with Computer. *Int. J. Theor. Phys.*, 21, 467 -- 488 (1982)
5. Feynman, R.P.: Quantum Mechanical Computers. *Found. Phys.*, 16, 507 -- 531 (1986)
6. Fortnow, L., Homer, S.: A Short History of Computational Complexity. *Bull. EATCS*, 80, 95 -- 133 (2003)
7. Gurevich, Y.: *Evolving Algebras 1993: Lipari Guide*, E. Börger (ed.) Specification and Validation Methods. Oxford University Press, 9 -- 36 (1995)
8. Gurevich, Y.: Sequential Abstract State Machines capture Sequential Algorithms. *ACM Trans. Comp. Logic*, 1, 77 -- 111 (2000)
9. Holevo, A.S.: *Probabilistic and Statistical Aspects of Quantum Theory*. North-Holland Publishing Company, Amsterdam (1982)
10. Kolmogorov, A.N., Uspensky, V.A.: On the Definition of an Algorithm. *AMS Transl.*, ser. 2, 21, 217 -- 245 (1963)
11. Manin, Yu.I.: *Computable and Uncomputable (Cybernetics)*, (in Russian). Sovetskoe radio, Moscow (1980)
12. Manin, Yu.I.: *Mathematics as metaphor: selected essays of Yuri I. Manin*. AMS (2007)
13. Nielsen, M.A. and Chuang, I.L.: *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge (2000)
14. Post, E.L.: Finite Combinatory Processes -- Formulation I. *J. Symb. Logics*, 1(3), 103 -- 105 (1936)
15. Turing, A.M.: On computable numbers, with an application to the Entscheidungsproblem. *Proc. Lond. Math. Soc.*, 2(42), 230 -- 265 (1936)
16. Turing, A.M.: Computability and λ -Definability. *J. Symb. Logics*, 2(4), 153 -- 163 (1937)
17. White, S.: A Brief History of Computing. <http://trillian.randomstuff.org.uk/stephen/history/>

Checking Inconsistencies in UML Design

Iryna Zaretska¹, Oleksandra Kulankhina¹, Hlib Mykhailenko¹, and Roman Kovalenko¹

¹V.N. Karazin Kharkiv National University, Kharkiv, Ukraine
zar@univer.kharkov.ua, mary.cauliflower@gmail.com,
tas.nix@gmail.com, kovalenkoo.roman@gmail.com

Abstract. The paper presents a simple method and software implementation for checking inconsistencies in UML design and the general method of UML design verification using its own model and first order predicate logic to specify relations between components of the design. Unlike various existing methods the proposed ones are focused mostly on cross-diagram inconsistencies and strong adhering to object-oriented principles. The model proposed for the general method is based on the unified graph representation of UML diagrams.

Keywords. Software design, object-oriented approach, UML, design model, verification.

Key Terms. SoftwareSystem, SoftwareComponent, Object, Model, VerificationProcess.

1 Introduction

Software design has become an increasingly important part of the software lifecycle due to the increasing complexity of software under construction.

The Unified Modeling Language (UML) is the de facto standard for modeling software systems. The UML supports a wide range of diagrams for modeling software. UML diagrams are independent but connected; their meta-model describes them under a common roof.

Detection of errors at the software design level allows reducing a great number of problems in the late stages of software development. There are several approaches for testing design models but they are mainly dealing with intra-diagram inconsistencies or using scripts for design execution.

In this paper model faults concerned with cross-diagram inconsistencies are considered. First, we present a simple method for detecting the cross-diagram inconsistencies in a UML design and its Java implementation. Then the general model of UML design for verification and refining purposes is introduced and discussed.

2 Related Work

2.1 OCL Constraints

The most common approach to set the rules of consistency for a UML model is to use Object Constraint Language (OCL) which is supported by the majority of UML CASE tools. In fact a lot of OCL constraints are embedded into the wide spread UML CASE tools in order to provide inconsistency verification of the model. The fact is that one can freely use OCL on the intra-diagram level but not on the cross-diagram level.

2.2 Critic Approach

A number of UML visual design tools provide model verification support including syntax checking and structural and consistency analysis. One of the components of such integrated support tools are critics.

There are various definitions of a design critic or a critic system in the literature. A critic can be considered as an intelligent user interface that evaluates a design made by a user and provides feedback to assist the user to improve the design. Generally critic tools detect potential problems, give advice and alternative solutions, and possibly automated or semi-automated design improvements to the end user.

Design critic tools have been used in design tools for various domains, including software engineering, design sketches, education, etc. Several studies report the benefits of applying design critic tools in software developments activities [1 – 8].

One of the critic tools is ArgoUML, an open source UML CASE tool. This tool supports the editing of UML notation diagrams and detects common errors made by software designers. For example, after placing a class in a class diagram, several critiques are displayed reminding the user that the class requires a better attribute name, needs operations, constructor and associations with other classes, and its class name needs to be capitalized. Thus, the user is helped to improve the design through the critiques. Java API is used to implement these features.

Other examples of the critic-based tools are ArchStudio3, SoftArch, DAISY, IDEA, ABCDE-Critic and AIR. These tools provide knowledge to architects, designers, and requirement engineers who lack specific understanding of the problem or solution domains. These critic tools all produce critiques that are specific to their problem domain. They use various approaches such as Java API, Prolog rules and knowledge bases, first-order production systems etc. to design and define critiques constraints. These tools have several limitations such as particular code or design language orientation or difficulties in customization requiring comprehension the critic's domain.

2.3 UML Design Execution

A number of approaches have been proposed to execute UML models [9 – 12]. Most of them use UML models to generate high level language code and execute the generated code.

Mellor and Balcer [11] for example use model compilers to support UML model execution. A set of domain and platform-specific model compilers are available commercially for realtime system modeling. At present the compilers cannot be extended to incorporate specific checks as the compiler source is not freely available for modifications.

Another technique for executing UML designs is to execute code that is generated from the model. Assuming that the code and model both contain the same information, executing the code is the same as executing the model.

Trung T. Dinh-Trong et al. [12] offer the systematic approach to testing UML designs based on a Java-like action language (JAL) used to transform the UML design under test into the executable form and then exercise them with generated inputs.

3 Cross-Diagram Inconsistencies

It is quite important to verify that the information about the model of the system at one UML diagram does not contradict to the information at the other UML diagram. We call such contradictions cross-diagram inconsistencies.

A UML design may contain different cross-diagram inconsistencies. Some of them are listed below.

1. An instance of the class A sends the message to the instance of the class B at the Sequence diagram, but the class B isn't visible for the class A at the Class diagram (Fig. 1).

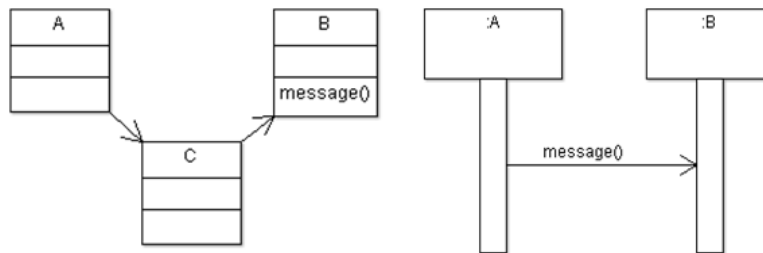


Fig. 1. Class B should be visible.

2. An instance of the class A sends the message to an instance of the class B at the Sequence diagram, but there is no corresponding method in the class B (Fig. 2).

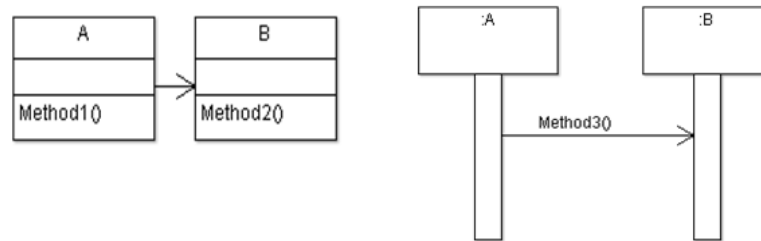


Fig. 2. Class B does not have method3().

- Transition from one state of the class A to another at the State Chart diagram occurs by the class A method invocation, but there is no such method in the class A at the Class diagram (Fig. 3).

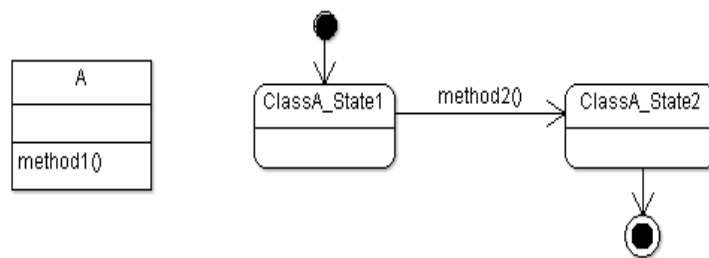


Fig. 3. Class A does not have method2().

- An instance of the class A sends the message to the class B (but not to the instance of this class) at the Sequence diagram, but the corresponding method of the class B isn't specified as static (Fig.4).

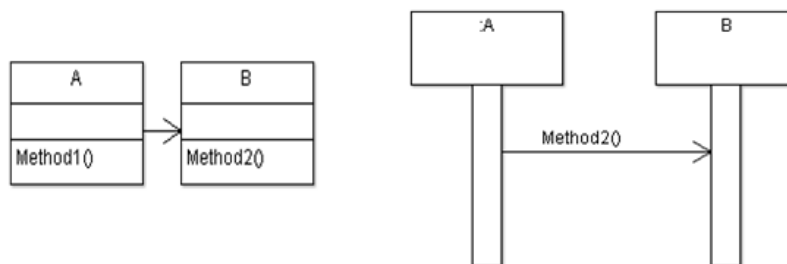


Fig. 4. In class B method2() is not specified as static.

As our analysis shows the most wide spread UML CASE tools cannot “see” such faults in the design and neither critic tools nor the design execution method are of any help in a cross-diagram verification process.

4 Simple Method for Detecting Cross-diagram Inconsistencies in UML Design

As most of the UML CASE tools allow exporting an object oriented design (OOD) of a target system into XMI format we offer a simple method of cross-diagram verification: we parse XMI file and find UML components dependences we are interested in. Depending on the type of an inconsistency under check we develop different check modules with their own models and consistency rules. The whole process looks like shown in Fig. 5. After parsing the XMI file into the Document Object Model (DOM) the Visitor takes care of getting over its elements and creating instances of the classes from the Checker’s model. Then the concrete Checker verifies this model according to its own rules and generates check results. To support this process the Java plug-in was developed. We call it Cross Diagram Inconsistency Check Plug-in (CDICP). It can be easily added to most of the UML CASE tools.

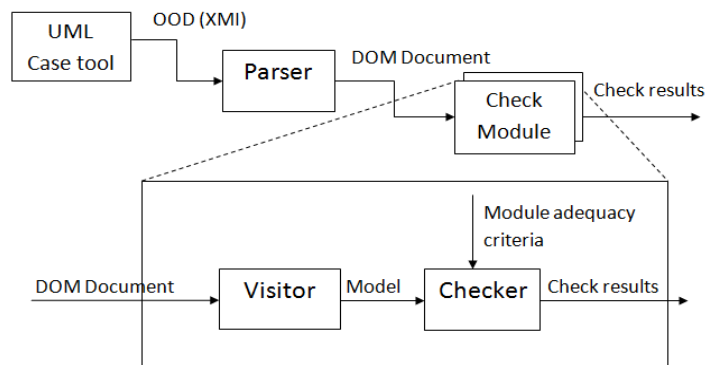


Fig. 5. The whole process of checking UML design.

As an example of a Check Module (Fig. 5) we developed Visibility Check Module (VCM) which finds the inconsistencies of the first type (Fig. 1).

VCM uses three classes for UML components representing; we call them Role, AssRole, and SeqRole (Fig. 6). The class Role represents a class at the Class diagram, the class SeqRole represents this class at the Sequence diagram (in fact they are two different UML components), and the class AssRole represents an association or dependency between classes at the Class diagram or a message between classes at the Sequence diagram (Fig. 7).

The Visitor in VCM identifies these components in the DOM, creates their instances and places them to the lists of classes, associations, messages, etc. Then the Checker applies its rules, verifies them and generates the result messages. The

Checker of VCM for every instance of the class AssRole, which represents some message between classes, checks if there is an association or dependency between these classes (another instance of the class AssRole) and detects its direction. If the connection is not found the Checker forms error message. This message contains information about cross-diagram inconsistency specifying its type and names of classes.

The CDICP plug-in for the VCM was developed and incorporated into Eclipse (Fig. 8).

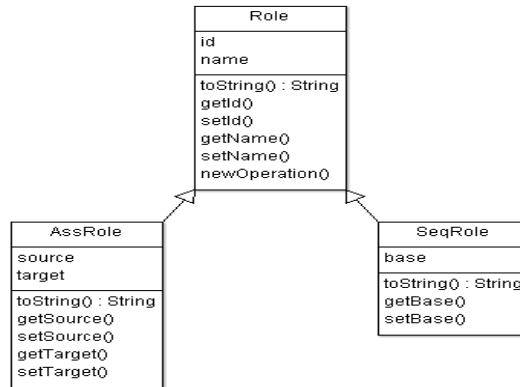


Fig. 6. Classes used by VCM.

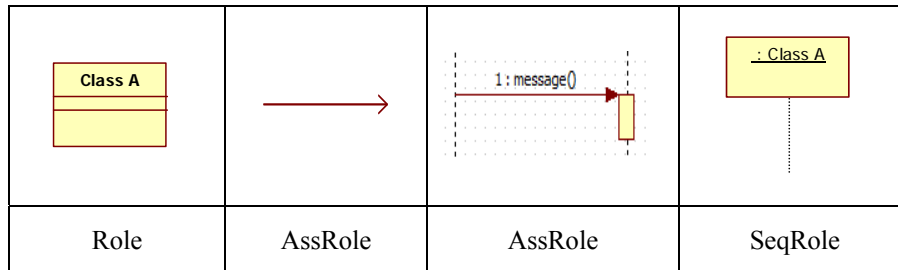


Fig. 7. Elements of UML design and correspondent classes of VCM.

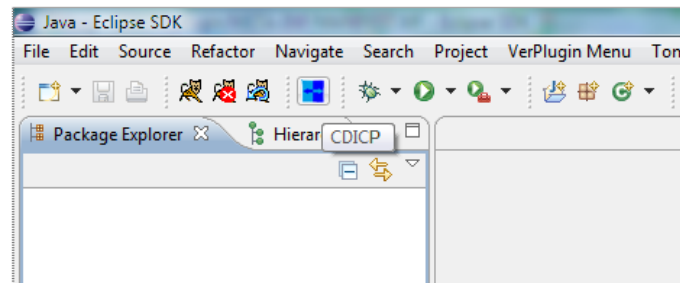


Fig. 8. New plugin CDICP in Eclipse.

5 General Method for Detecting Cross-diagram Inconsistencies in UML Design

Analyzing only four most important at the design stage diagrams which are Class diagram, Sequence diagram, Object Diagram and State Machine diagram (in UML 2.0 specification) [13, 14] we defined more than 30 intra- and cross-diagram relations to be checked. None of well known CASE tools offers such checks. Using the simple method mentioned above is quite tedious as it supposes developing a special Checker (Fig. 8) for each relation, which in its turn requires multiple searches on XMI file. Even for a middle size project this file is quite big. The main idea here is to develop a special model of the system design for verification purposes (as usually done in verification methods), build it once by parsing XMI file, and then make all checks on this model. Moreover such model can be used for refining design on account of lessening couplings, strengthening cohesion and applying design patterns. All results of this model analysis regardless of the purpose take the form of recommendations so the corrective changes are up to the designer.

Thorough analysis of UML 2.0 specification led us to using graph representation of such model. It allows unified representation of all four diagrams by graphs with different types of vertices and edges. In this case checking relations between UML diagrams is just searching for the definite types of vertices or edges or their interconnections in the model. The first order predicate logic is used to formulate the relations leading to inconsistencies. In fact graph representation simplifies the description of diagrams comparing to their formal specification but is sufficient for verification purposes. For a class diagram the corresponding graph's vertices are classes and edges are connections between them which are association, dependency, generalization and interface realization. The information about generalization sets is stored separately to simplify search algorithms. For an object diagram the corresponding graph's vertices are objects and edges are connections between them. For a sequence diagram the vertices are objects and edges are messages between them. For a state machine (or state chart) diagram the vertices are states and edges are transitions between them. Each type of vertex and each type of edge stores information needed to check intra- and cross-diagram inconsistencies. Say an association of a class diagram as an edge of a graph keeps the name of the association, roles and multiplicities of its participants, etc. An example of the simple class diagram and its graph representation is given in Fig. 9. The edges of the graph represent different types of connections between classes and hence store different information.

Here is the formal representation of our model consisting of graphs of four types for Class, Object, Sequence and State Machine diagrams correspondently:

$$D = \{\{D_{cl}\} \cup \{D_{ob}\} \cup \{D_{seq}\} \cup \{D_{st}\}\}.$$

Each of these graphs consists of two sets: V stands for vertices and E stands for edges. Their description is given below.¹

$$D_{cl} = \left\{ V_{cl}, E_{cl} \right\}$$

¹ Elements in [] are optional.

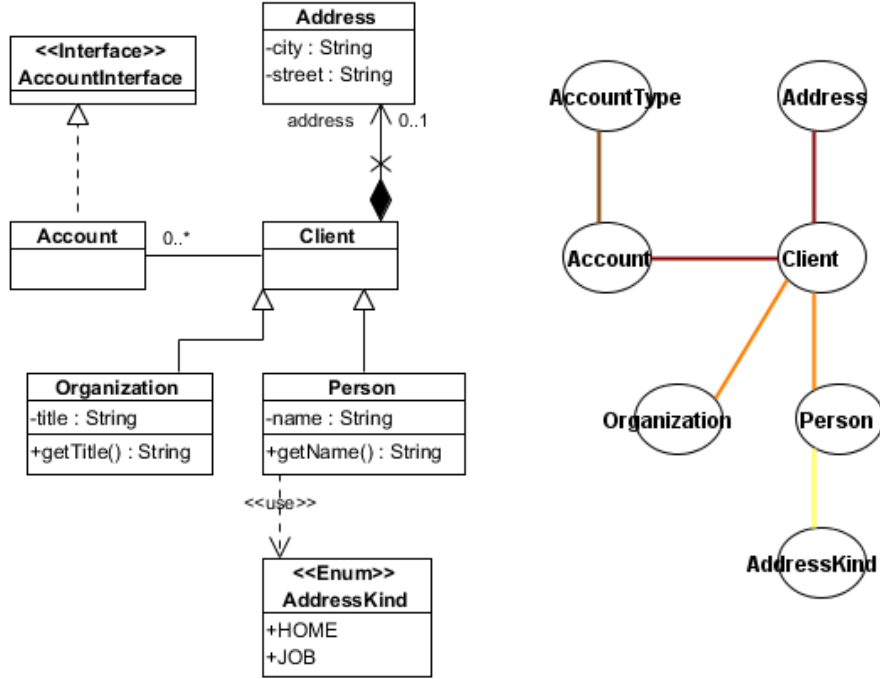


Fig. 9. Example of graph representation of a class diagram

$$\begin{aligned}
 V_{cl} &= \{v : v = (\text{name}, \text{isAbstract}, \text{ATTR}, \text{MTHD}, \text{STRT}, \text{visibility})\} \\
 \text{ATTR} &= \{\text{attr} : \text{attr} = (\text{name}, \text{domain}, \text{scope}, \text{visibility}, \text{multiplicity})\} \\
 \text{MTHD} &= \{\text{mthd} : \text{mthd} = (\text{mthdSgn}, \text{scope}, \text{visibility})\} \\
 \text{mthdSgn} &= (\text{name}, \text{PARAMS}, \text{returnDomain}) \\
 \text{PARAMS} &= \{\text{param} : \text{param} = (\text{num}, \text{name}, \text{domain})\} \\
 \text{STRT} &= \{\text{stereotype} : \text{stereotype} = (\text{name})\} \\
 E_{cl} &= \{e : e = (v_s, v_e, \text{type}, \text{info}); v_s, v_e \in V_{cl}, \text{type} = \text{gen} | \text{ass} | \text{dep} | \text{impl}\} \\
 \text{info} &= ([\text{name}, r_s, r_e, m_s, m_e, \text{aggr}_s, \text{aggr}_e, \text{navig}_s, \text{navig}_e]) \\
 D_{ob} &= \{V_{ob}, E_{link}\} \\
 V_{ob} &= \{v : v = (\text{name}, \text{clName}, \text{ATTRVAL}, \text{STRT})\} \\
 \text{ATTRVAL} &= \{\text{attrval} : \text{attrval} = (\text{name}, \text{value})\} \\
 E_{link} &= \{e : e = (v_s, v_e, \text{name}); v_s, v_e \in V_{ob}\} \\
 D_{seq} &= \{V_{cl} \cup V_{ob}, E_{msg}\} \\
 E_{msg} &= \{e : e = (v_s, v_e, \text{msgCall}); v_s, v_e \in V_{cl} \cup V_{ob}\} \\
 \text{msgCall} &= ([\text{guard}, \text{seqnum}, \text{mthdCall})
 \end{aligned}$$

$$\begin{aligned}
 mthdCall &= (name, ARGS[, returnValue]) \\
 ARGS &= \{armnt : armnt = (num, value)\} \\
 D_{st} &= \{V_{st}, E_{tr}\} \\
 V_{st} &= \{v : v = (name, [, entry, do, exit]); entry, do, exit \in mthdCall\} \\
 E_{tr} &= \{e : e = (v_s, v_e, trCall); v_s, v_e \in V_{st}\} \\
 trCall &= ([guard,] mthdCall).
 \end{aligned}$$

An example in Fig. 10 illustrates information stored with some types of vertices and edges.

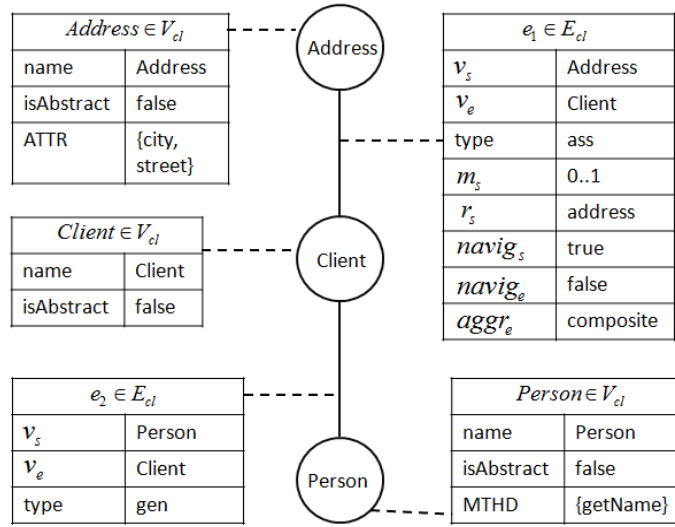


Fig.10. Information stored in graph elements for the example above

The relation to be checked should be represented as the first order predicate logic formula. Propositional variables in this formula are the elements of the model above. Such unified approach to formulating the criteria of relations to be checked allows using the only Checker for any sort of relation.

Here is an example of such formula. It describes the fact that if the instance of one class sends the message to the instance of another class in the sequence diagram then the corresponding method should be among the methods of the latter class in the class diagram (Fig. 2 shows an example of this relation not satisfied).

$$\begin{aligned}
 &(\forall e \in E_{msg} : v_e(e) \in V_{cl}) \\
 &(\exists v \in genPath(v_e(e)))(\exists mthd \in MTHD(v(e))) : msgCall(e) \approx mthdSgn(mthd) \\
 &\wedge \\
 &(\forall e \in E_{msg} : v_e(e) \in V_{ob}) \\
 &(\exists cl \in V_{cl})(\exists v \in genPath(v_e(e)))(\exists mthd \in MTHD(v(e))) : msgCall(e) \approx mthdSgn(mthd)
 \end{aligned}$$

where

$$genPath(v) = v_1 \dots v_n : v_1 = v \wedge (\forall i = 1, n-1)(\exists e \in E_{cl} : type(e) = gen \wedge v_s(e) = v_i \wedge v_e(e) = v_{i+1})$$

is introduced to take into account the fact that the method in question can be inherited along the path in the inheritance tree of the class.

At the moment the software tool for the proposed method is being developed and tested.

6 Conclusions

This paper offers a simple method for detecting inconsistencies between different UML diagrams. It was implemented as an Eclipse plug-in and tested on some types of cross-diagram inconsistencies.

Another more general method for checking inconsistencies in UML design is proposed. It uses the unified model with graph representation of the design components and formulae of the first order predicate logic to represent relations which should be satisfied to make the design consistent. This approach can also be used to evaluate the quality of a design and make recommendations on its improvement on account of better use of the main principles of the object-oriented design.

References

1. A. Andrews, R. B. France, S. Ghosh, and G. Craig.: Test Adequacy Criteria for UML Design Models. *Journal of Software Testing, Verification and Reliability*, 13(2), pp. 95--127 (2003)
2. Fischer. G. et al.: The Role of Critiquing in Cooperative Problem Solving, *ACM Transactions of Information Systems*, Vol.9, No.3, pp. 123--151 (1999)
3. Lionel Briand and Yvan Labiche: A UML-based approach to system testing. *Software and System Modeling*, 1(1), pp. 10--42 (2004)
4. Souza, C.R.B., et al.: Using Critiquing Systems for Inconsistency Detection in Software Engineering Models. In: *Proceedings of the Fifteenth International Conference on Software Engineering and Knowledge Engineering (SEKE 2003)*, San Francisco Bay, pp. 196--203 (2003)
5. Souza, C.R.B., et al.: A Group Critic System for Object-Oriented Analysis and Design. In: *Proceedings of the 15th IEEE International Conference on Automated Software Engineering (ASE'2000)*, pp. 313--316 (2000)
6. S. Ghosh, R. B. France, C. Braganza, N. Kawane, A. Andrews, and O. Pilskalns: Test Adequacy Assessment for UML Design Model Testing. In: *Proceedings of the International Symposium on Software Reliability Engineering*, pp. 332--343, Denver, CO (2003)
7. Mara del Mar Gallardo, Pedro Merino, Ernesto Pimentelis: Debugging UML Designs with Model Checking. *Journal of Object Technology*, 1(2), pp. 101--117 (2002)
8. Martin Gogolla, Jrn Bohling, and Mark Richters: Validation of UML and OCL models by automatic snapshot generation. In: *Proceedings of the 6th International Conference on Unified Modeling Language (UML'2003)*, pp. 265--279. Springer, Berlin, LNCS 2863 (2003)

9. Nilesh Kawane: Fault Detection Effectiveness of UML Design, Model Test Adequacy Criteria. In: Supplementary Proceedings of the International Symposium on Software Reliability Engineering, pp. 327--328, Denver, CO (2003)
10. Nilesh Kawane: EPTUD : An Eclipse plug-in for testing UML design models. Master's of science thesis, Colorado State University, Fort Collins, Colorado (2005)
11. Stephen Mellor and Marc Balcer: Executable UML: A Foundation for Model Driven Architecture. Addison Wesley Professional (2002)
12. T. Dinh-Trong, N. Kawane, S. Ghosh, R. B. France, and A. A. Andrews: A Tool-Supported Approach to Testing UML Design Models. In: 10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2005), Shanghai, China, Proceedings (2005)
13. Object Management Group: UML 2.0 Superstructure Specification (2005), <http://www.uml.org/>
14. Pender. T.: UML Bible. Wiley Published Inc. (2003)

Automatic Tests and Practical Tasks Generation in Distance Learning Systems

Dmytro Kravtsov¹

¹Kherson State University
hwndmaster@gmail.com

Abstract. This article deals with the problem of the developing and implementation of the tests and practical tasks generator in the distance learning systems. The object model of the tests and exercises generator is developed based on the mathematical model.

Keywords: Smart-test, mathematical test, automatic tests building, remote testing, remote practical tasks.

Key Terms: InformationTechnology, KnowledgeEngineeringMethodology, Development, ModelBasedSoftwareDevelopmentMethodology.

1 Introduction

The software module that support practical tasks and academic performance ratings during the learning process are the most important in distance learning systems (DLS) [1]. A test is the major element of the academic performance ratings in DLS. Generally, tests consist of the tasks (questions) that deal with the all topics of the studied subject matter. Practical tasks and tests' questions in some sciences, more often exact sciences such as mathematics, physics, chemistry, informatics, etc., are characterized with the similar wordings that may be shown with templates. Tasks templates are the formal expressions that are described with the strict syntax and have well-defined semantics. These expressions contain parameters with the exact given range of values. A task instance is created with the substitution of the specific parameters of the defined area of values into the sample.

The practical tasks and tests development is rather hard and tedious process of tasks (questions) creation, possible answers defining, and correct answers detection [2]. The following drawbacks of the test modules are typical for the majority of the wide-used DLS:

- time consuming process of tasks (questions) compilation;
- requirement to develop a wide variety of similar tasks to provide students with representative sets of questions or to solve a problem of students copying the each others tasks solving;

— possible mistakes committed by a tasks compiler.

The good example of the software implementation for this type of tasks is Generator of examination tests in software environment «Examination tests delivery system» [3].

2 Problem Definition

While DLS testing module engineering, the process of testing is usually understood as a set of statements calls with a certain behavior in the context of a learning model. The process of the automatic test compiling in DLS testing module is a work flow in which a certain logics with the possibility of automation is implemented. It is important to start with a determination of an abstract model, that contains all the necessary data describing the learning aim and explaining the correlation between objects of the specific instances of this model, to automate the compiling of tests' questions as a work flow.

Let's say *smart-test* is the instance of the test, generated by the given template with the automatic test generation process that is described above.

The automatic generation of a smart-test will help to substantially simplify the practical tasks compilation in the mentioned above sciences, reducing required time and guaranteeing the unique character of each task generated, as well as it gives the ability to automate the tests results evaluation without negative human factor effect.

In this work we consider the problem of creation of a smart-test generator abstract model and tasks for solving the most of the practical tasks of the applied disciplines, and the implementation work of the smart-tests generator in DLS. The abstract generator model is a generalized model of *mathematic tests* [4]. The results of the work will be illustrates with the examples of the specific mathematics discipline tasks.

3 Abstract Model

We define the smart-test as a set of learning tasks set and test settings: $ST = \langle S(s_1, \dots, s_L), P(P_1, \dots, P_N) \rangle$, where $S(s_1, \dots, s_L)$ – a set of test settings, defined by a user (a test compiler), and $P(P_1, \dots, P_N)$ – a set of learning tasks, that define the term *mathematic tests* [4].

To build the P – abstract model of the learning task – we use its mathematical definition [4], that has the following view:

$$P = \langle M, \Phi, Q \rangle,$$

where

M – a set of the task input parameters, and is described as $M(x_1, \dots, x_n)$, where x_1, \dots, x_n – the task parameters;

Φ – a set of the task conditions, described as $\Phi(\varphi_1, \dots, \varphi_k)$, where $\varphi_1, \dots, \varphi_k$ – task conditions;

Q – the task result model, described as $Q(q_1, \dots, q_m)$, where q_1, \dots, q_m – sets of correlations, defining the results of the task.

This mathematic model is sufficient for further abstractions building and object-realization with the high-level programming languages.

Let's agree with the following terms to describe the mathematic model above in a programming language:

$P = \langle M, \Phi, Q \rangle$ – LrnTaskTemplate (abbreviated from Learning Task Template);

M – LrnTaskModel (abbreviated from Learning Task Model);

Φ – LrnTaskValidator (abbreviated from Learning Task Validator);

Q – TResponse, that is the element of the software module «Solver» LrnTaskSolver<TModel, TResponse>, that will be described below.

The interrelations scheme of the objects described above is introduced at fig.1.

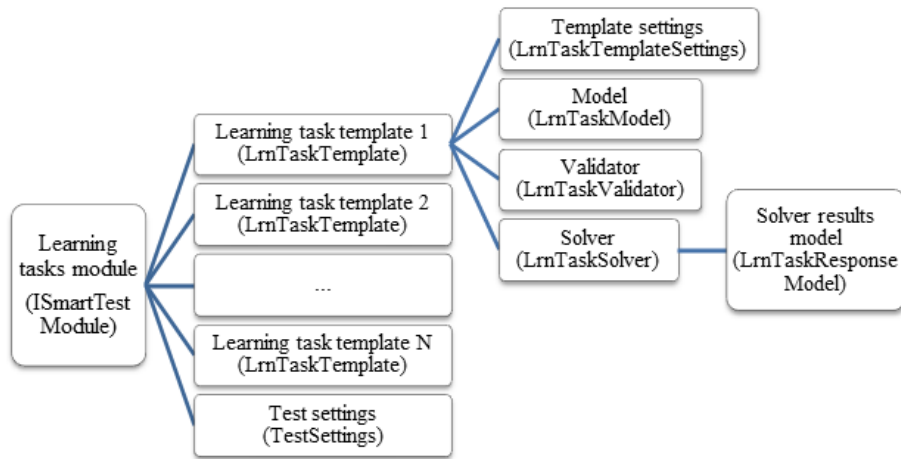


Fig. 1. Smart-test object model elements interrelations scheme.

On this scheme ISmartTestModule is a template library of the learning tasks generators for smart-tests compilation.

A *template library of the learning tasks generator module* is a program class (LrnTaskTemplate), which contains all the functions necessary for building the learning task instances, their correction, and solvings with the use of named model.

Task generator template functions use general (TestSettings) and local (LrnTaskTemplateSettings) settings, given by a user (test compiler) as input parameters. These settings may provide the template with generation and validation parameters of the specified learning tasks as: instruction to use preferable types of questions [7], number of answers to generate, module fine-tuning with including/excluding certain templates, field of input parameters, etc. The number of such settings may differ depending on the module and its templates. It is expected that the test compiler tune the settings while adding a smart-test to the system.

Learning task model (LrnTaskModel) is the program class that aim to store the input parameters of the learning task. Depending on the specified aims, the learning

task model may contain advanced parameters, necessary to describe the task settings. The number of the parameters is not limited by the system.

The next learning tasks generator element is `LrnTaskValidator` class. The aim of this program class is *validation* of the model instance to correspondence to the task settings. This program class is also used in reconciliation of the generated learning task model that is described further.

A *solver* (`LrnTaskSolver`) is the final element of the template. The aim of the solver is to solve the specified task (achieve results), using the `LrnTaskModel` parameters with the conditions given in `LrnTaskValidator`. The results of the solver are the formal expressions in the signature of the certain subject field that may be defined in the `LrnTaskResponseModel` class inheritor. In particular, atomic expressions may be there – a number, a string, a date.

4 The Model of the Smart-test Implementation Software Module

The model of the learning tasks generator template module is shown at fig. 2.

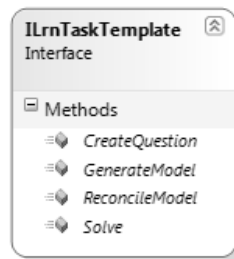


Fig. 2. The model of the learning tasks generator template module.

The list of the main *functions of the tasks generator* is below:

1. `GenerateModel` is a method of learning task model instance creation (`LrnTaskModel`).
2. `ReconcileModel` is a method of generated learning task reconciliation. This function is necessary to solve the problem, when the system generates similar tasks at the first stage of task generation in the set of the final smart-test. The correction algorithm must be implemented locally for each type of the learning tasks. The aim of the correction is to change the learning task model by making it unique comparing with other tasks in the set of the tasks in the smart-test.
3. `CreateQuestion` is a method of the question description creation. This method creates a program class instance that describes the question. This class model is shown at fig. 3:

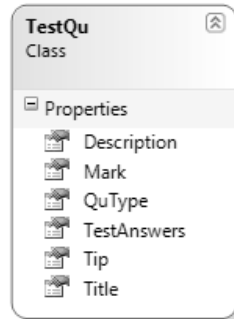


Fig. 3. Base model of the program class containing question description.

Here is the description of *model properties*:

- Title - title-text of the task. This property consists of the short text, that explains students the subject of the question;
 - Description - the description of the task that will be shown to the student. An author of the specified learning tasks module implementation should consider possibility to create description text on one of the system languages (for example, Russian, Ukrainian, English);
 - Tip - a hint text, that the student may use answering the question. This field is optional and may be left empty. Usage of the tip is fixed and processed by the system;
 - QuType - type of the question. An author of the specified learning tasks module implementation may use one of the multiple questions types, mentioned in the specifications [7];
 - Mark - a grade that a system assigns for the correct answer for the question.
4. Solve - a method of the task solving. This function uses the learning task model (LrnTaskModel) as the input parameter and solves it, using the functions of the LrnTaskSolver program class. The results of the solving returns as the object, consisting of the correct answer for this learning task.

5 Examples of Smart-tests Implementation in DLS Kherson Virtual University [5]

Here is the example of learning tasks templates implementation on “Quadratic equation” topic [4, 6]. Below you can see the objects implementation, which inherit properties and methods of the abstractions described above:

1. QuadraticEquationTestModule learning tasks module, functions and properties of the program interface ISmartTestModule (fig.4):

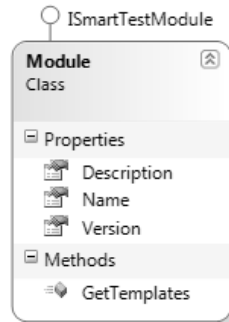


Fig. 4. QuadraticEquationTestModule program class.

2. Template 1 - QuadraticEquationTestTemplate1, that implements functions and properties of the program interface ILrnTaskTemplate. This template will generate tasks with the title: “Solve the quadratic equation with integer coefficients”;
3. Template 2 - QuadraticEquationTestTemplate2, that implements functions and properties of the program interface ILrnTaskTemplate. This template will generate tasks with the title: “Find the sum of the squares of the roots of a given quadratic equation” (fig. 5):

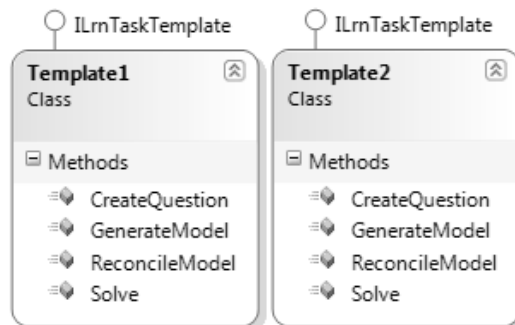


Fig. 5. QuadraticEquationTestTemplate1 and QuadraticEquationTestTemplate2 program classes.

4. QuadraticEquationLrnTaskModel implements the functions of the abstract program class LrnTaskModel (fig. 6). This model is actual for the both templates:

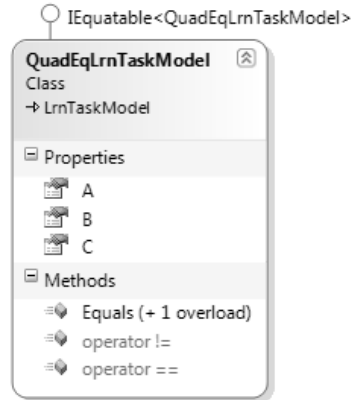


Fig. 6. QuadraticEquationLrnTaskModel program class.

5. QuadraticEquationLrnTaskValidator implements the functions and properties of the abstract program class LrnTaskValidator (fig. 7). This validator is actual for the both templates:

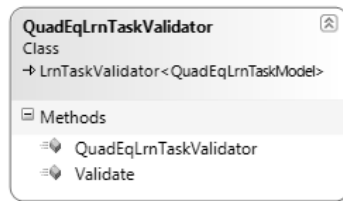


Fig. 7. QuadraticEquationLrnTaskValidator program class.

6. Template 1 Solver - QuadraticEquationLrnTaskSolver implements functions and properties of the abstract program class LrnTaskSolver (fig. 8):

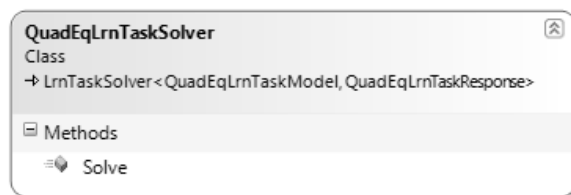


Fig. 8. QuadraticEquationLrnTaskSolver program class.

Executing the solution from the parameters of the mentioned model, this solver returns the object of the QuadraticEquationLrnTaskResponse type (fig. 9), that contains a pair of numerical values: x_1 , x_2 :

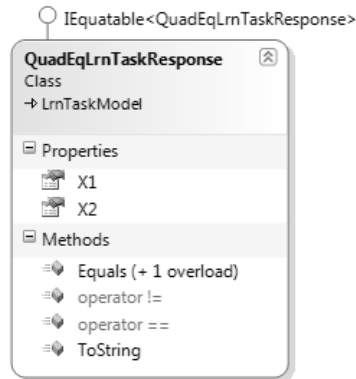


Fig. 9. QuadraticEquationLrnTaskResponse program class.

7. Template 2 solver - `QuadraticEquationLrnTaskSolverU`, which implements functions and properties of the abstract program class `LrnTaskSolver` (fig. 10). Executing the solution from the parameters of the mentioned model, this solver returns one numerical value x :

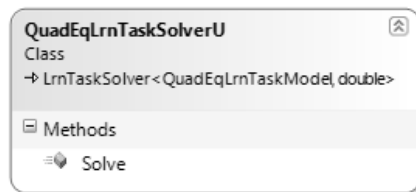


Fig. 10. QuadraticEquationLrnTaskSolverU program class.

8. Test setting class - `QuadraticEquationTestSettings` implementing the program interface `ILrnTaskTemplateSettings` (fig. 11):

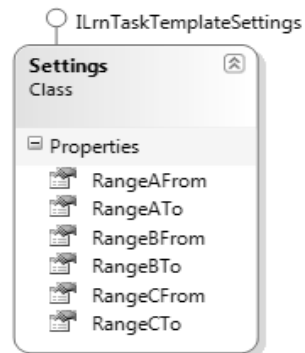


Fig. 11. QuadraticEquationTestSettings program class.

In this example it is supposed to specify valid values in the test settings for quadratic equation coefficients: a , b and c .

Now let us turn to the algorithm of creating smart-test and learning tasks using the object model, mentioned above.

The process of the smart-test creation begins with the connecting necessary modules to the system interface. We should mention that in this example the user has just one module: QuadraticEquationTest. The user should point in the module settings the templates to use (there are only two in this example – Template 1 and Template 2), and the settings of each of the template. The test compiler also sets number of questions: N , which should be generated. One of the parameters, that the user should specify, is the parameter of the rate set, which specifies the coefficients of usings of module templates in the smart-test.

When all the settings are made by the user, a smart-test is triggered after each request of a student in the testing group of the distance learning system. Below you can see the algorithm of the test instance generation (fig. 12).

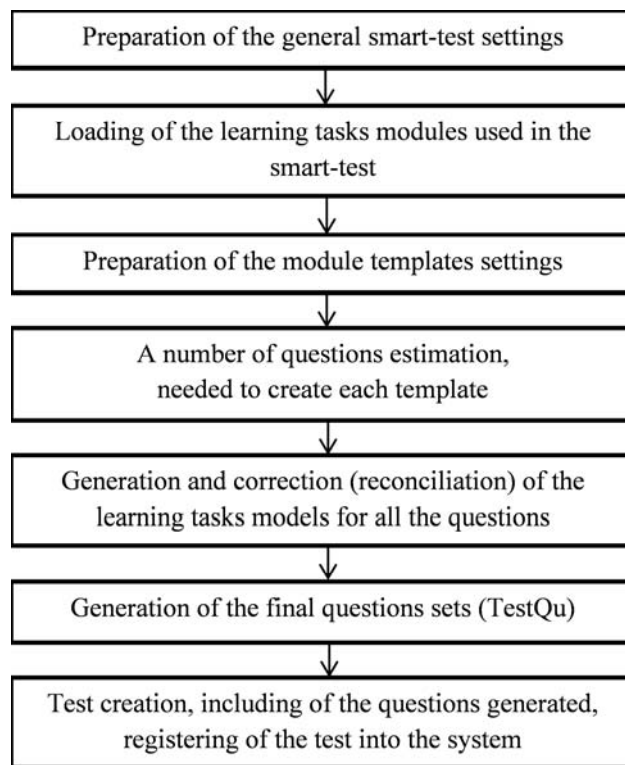


Fig. 12. Algorithm of the final test instance generation scheme.

A number of questions estimation. Assume N - a number of questions, needed to generate the test, $T = \langle t_1, \dots, t_k, q_1, \dots, q_k, s_1, \dots, s_k \rangle$ - a set of used templates from all the connected modules, where t_i – the instance of template class, q_i – rate of using of i -template to the all number of templates T , s_i – template settings, made by the test compiler.

Example. Assume $N = 20$, $T = \langle t_1, t_2, 0.3, 0.7, s_1, s_2 \rangle$.

Then the final test will consist of 20 questions, based on the templates t_1 and t_2 . 30% of the questions (6 questions) will be generated based on the first template. 70% of the questions (14 questions) will be generated based on the second template.

Learning tasks modules creation. Below you can see the algorithm scheme of the learning task module creation and correction (fig. 13).

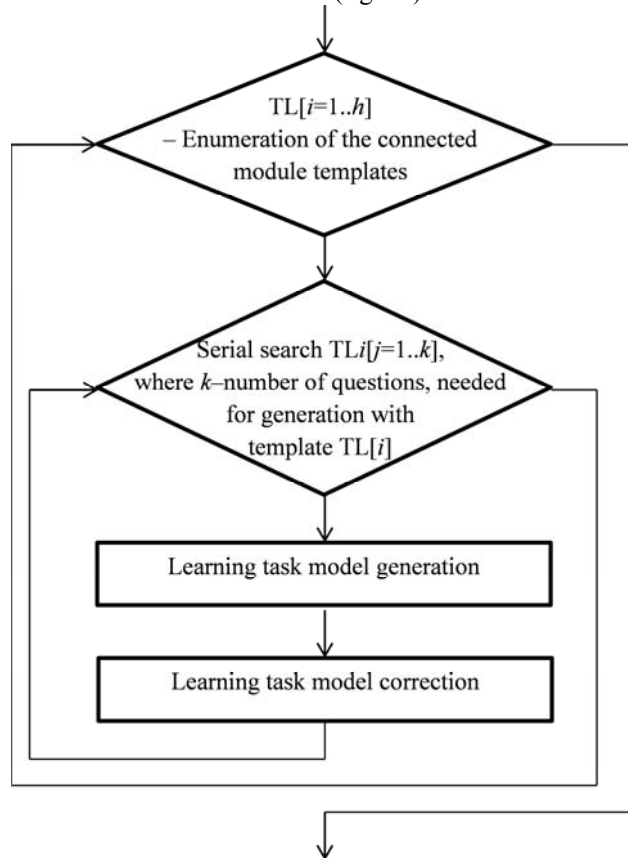


Fig. 13. Scheme of the sub-program algorithm of the learning task model generation and correction.

As one can see from the scheme above, model correction is implemented immediately after it is generated. It allows, first of all, using rough method of the model values generation, using the system random number generator. Then in the correction function the model parameters are changed not to coincide with the other models, generated in the cycle before.

Generation of the final questions sets (TestQu). The process of the final set of questions creation, based on the already generated and adjusted ones, occurs with the same enumeration as a generation of the learning tasks models - first all the templates of the connected modules are enumerated, and then the number of this template questions is searched through.

The sub-program of the learning task model question instance creation is shown at fig. 14.

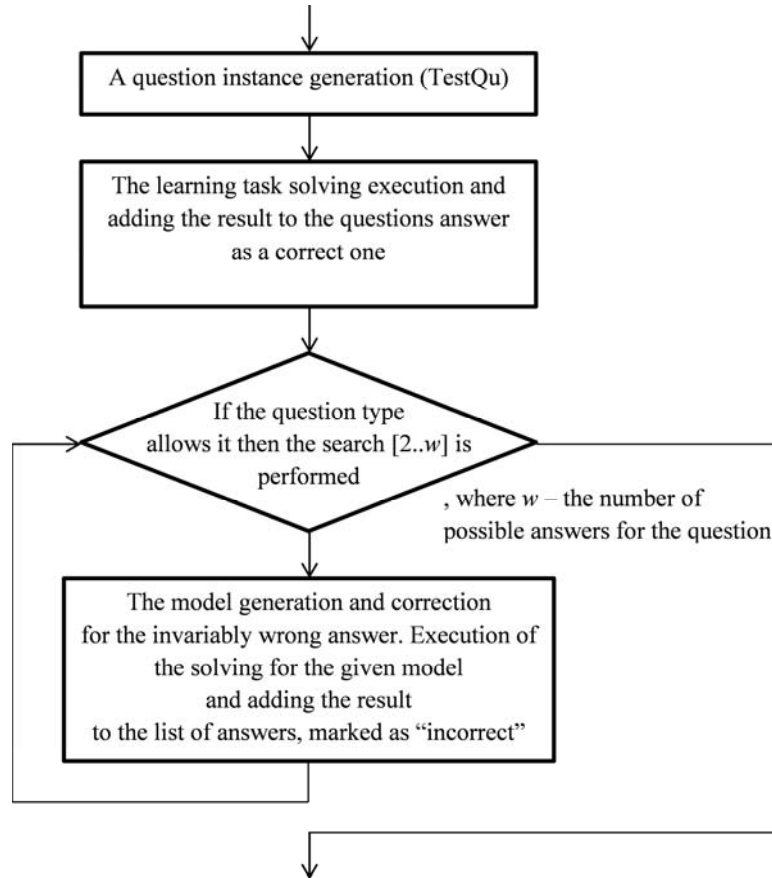


Fig. 14. The question instance generation sub-program algorithm scheme.

Thereby, smart-test is initialized and registers in the distance learning system, and then it becomes ready for students to pass it.

6 Conclusion

The structural model of the software module for the smart-tests generation and use, based on the method of automatic generation with the assigned tests or practical tasks template is build on the basis of mathematical model.

The software module classes, algorithms of learning tasks and questions models generation and verification, and smart-test answers processing are described. The described model was tried and tested on the examples of the smart-test for the exact

sciences: mathematics, physics, chemistry, computer science. Also it seems possible to make smart-tests in the field of some other disciplines: some biology sections, psychology, music, sociology, etc.

This method of automatic test generation, in the long view, may be used while creation of adaptive tests with the use of smart-tests.

References

1. Bykov V. Yu.: Models of the Open Education Organizational Systems: Monograph. – Kyiv: Atika, (2009) – p. 684. (In Russian)
2. H. Kravtsov, D. Kravtsov.: Knowledge Control Model of Distance Learning System on IMS Standard / Innovative Techniques in Instruction Technology, E-learning, E-assessment, and Education. – Springer Science + Business Media V.B. pp.195 – 198 (2008) (In Russian)
3. Kruchinin V.V., Magazinnikov L.I., Morozova Yu.V.: Control tasks generator, VIII International scientific conference and exhibition “Unified educational information environment: problems and development trends”, Tomsk, 17 (2009), http://www.ict.edu.ru/vconf/index.php?a=vconf&c=getForm&r=secDesc&d=mod&id_vconf=30&id_sec=174. (In Russian)
4. Lvov M.S.: Mathematical tests in the systems of computer mathematics of the educational purpose. / M.S. Lvov // Control systems and machines. - 2011. - №6. (In Russian)
5. Distance learning system Kherson Virtual University, <http://dls.kherson.ua/dls>.
6. Quadratic equation, the Free Encyclopedia, Wikipedia, http://en.wikipedia.org/wiki/Quadratic_equation.
7. IMS Question & Test Interoperability Specification, <http://www.imsglobal.org/question/>

Satisfiability Problem in Composition-Nominative Logics of Quantifier-Equational Level

Mykola S. Nikitchenko¹ and Valentyn G. Tymofieiev¹

¹Department of Theory and Technology of Programming
Taras Shevchenko National University of Kyiv
64, Volodymyrska Street, 01601 Kyiv, Ukraine
nikitchenko@unicyb.kiev.ua
tvalentyn@univ.kiev.ua

Abstract. We investigate algorithms for solving the satisfiability problem in composition-nominative logics of quantifier-equational level. These logics are algebra-based logics of partial predicates constructed in a semantic-syntactic style on the methodological basis, which is common with programming; they can be considered as generalizations of traditional logics on classes of partial predicates that do not have fixed arity. We show the reduction of the problem in hand to the satisfiability problem for classical first-order predicate logic with equality. The proposed reduction requires extension of logic language and logic models with an infinite number of unessential variables. The method developed in the paper enables us to use existent satisfiability checking procedures also for quantifier composition-nominative logic with equality.

Keywords: Composition-nominative logics, partial predicates, partial logics, first-order logics, satisfiability, validity.

Key Terms. Research, MathematicalModel, FormalMethods, MachineIntelligence.

1 Introduction

Last years the interest to the satisfiability problem [1] has risen due to practical value it has obtained in such areas as program verification, synthesis, analysis, testing, etc. [2–5]. In this paper we address the satisfiability problem in the context of the *composition-nominative approach* [6], which aims to construct a hierarchy of logics of various abstraction and generality levels on the methodological basis, which is common with programming. The main principles of the approach are principles of *development from abstract to concrete*, *priority of semantics*, *compositionality*, and *nominativity*.

These principles specify a hierarchy of new logics that are semantically based on algebras of predicates. Predicates are considered as partial mappings from a certain class of data D into the class of Boolean values $Bool$. Operations over predicates are called *compositions*. They are treated as predicate construction tools. Data classes are considered on various abstraction levels, but the main attention is paid to the class of

nominative data. Such data consist of pairs *name–value*. Nominative data can represent various data structures such as records, arrays, lists, relations, etc. [6, 7]; this fact explains the importance of the notion of nominative data. In the simplest case nominative data can be considered as partial mappings from a certain set of names (variables) V into a set of basic (atomic) values A . These data are called *nominative sets*; their class is denoted ${}^V A$. Nominative sets represent program states for simple programming languages (see, for example, [6, 8]). Partial predicates and functions over ${}^V A$ are called *quasiary*, their classes are denoted $Pr^{V,A} = {}^V A \xrightarrow{p} Bool$ and $Fn^{V,A} = {}^V A \xrightarrow{p} A$ respectively. Partial mappings of type ${}^V A \xrightarrow{p} {}^V A$ are called *bi-quasiary*. Such mappings represent program semantics for simple programming languages; therefore their class is denoted $Pr_g^{V,A}$. From this follows that semantic models of programs and logics are mathematically based on the notion of nominative set (nominative data in general case). This fact permits to integrate models of programs and logics and represent them as hierarchy of composition-nominative models [9, 10]. Logics developed within such approach are called *composition-nominative logics* (CNL) because their predicates and functions are defined on classes of nominative data, and logical connectives and quantifiers are formalized as predicate compositions.

CNL can be considered as generalization of classical predicate logic but for all that many methods developed within classical logic can also be applied to CNL. Here we confirm this statement for the satisfiability problem in CNL. In this paper we consider composition-nominative logic of quantifier-equational level and construct an algorithm that reduces the satisfiability problem in this logic to the same problem in classical first-order predicate logic with equality. The reduction proposed requires the logic language to be extended with an infinite number of unessential variables.

The paper is structured in the following way. In section 2 we give an overview of the composition-nominative logics classification; then in section 3 we give formal definitions of the logics that we consider in this paper, and define the satisfiability problem. In section 4 we describe the reduction method for solving the satisfiability problem. In section 5 we discuss related work. In section 6 we summarize our results and formulate directions for future investigations.

Proofs are omitted here and will be provided in an extended version of the paper.

Notions and notations not defined in the paper are understood in the sense of [10].

2 Classification of Composition-Nominative Logics

Classification of *composition-nominative logics* is based on classification of their parameters: data, predicates, and compositions. The main semantic notion of mathematical logic – the notion of predicate – can be defined as a partial function from a data class D to $Bool$. For the most abstract level of data consideration such compositions as disjunction \vee , negation \neg , etc., can be defined. These compositions are derived from Kleene’s strong connectives [11] when partiality of predicates is taken into consideration. Thus, the main semantic objects for logics of this level are algebras of partial predicates of the type $\langle D \xrightarrow{p} Bool; \vee, \neg \rangle$. The obtained logics may be

called *propositional logics of partial predicates*. Such logics are rather abstract, therefore their further development is required at the nominative level. At this level we have two sublevels determined respectively by flat and hierarchic nominative data.

Three kinds of logics can be constructed from program models on the flat nominative data level:

1. pure quasiary predicate logics based on algebras with one sort: $Pr^{V,A}$;
2. quasiary predicate-function logics based on algebras with two sorts: $Pr^{V,A}$ and $Fn^{V,A}$;
3. quasiary program logics based on algebras with three sorts: $Pr^{V,A}$, $Fn^{V,A}$, and $Prg^{V,A}$.

For logics of pure quasiary predicates we identify renominative, quantifier, and quantifier-equational levels.

Renominative logics [10] are most abstract among the above-mentioned logics. The main composition for these logics is the composition of renomination (renaming), which is a total mapping $R_{x_1, \dots, x_n}^{v_1, \dots, v_n} : Pr^{V,A} \xrightarrow{t} Pr^{V,A}$. Intuitively, given a quasiary

predicate P and a nominative set d , the value of $R_{x_1, \dots, x_n}^{v_1, \dots, v_n}(P)(d)$ is evaluated in the following way: first, a new nominative set d' is constructed from d by changing the values of the names v_1, \dots, v_n in d to the values of the names x_1, \dots, x_n respectively; then predicate P is applied to d' . The obtained value of P (if it was evaluated) will be the result of $R_{x_1, \dots, x_n}^{v_1, \dots, v_n}(P)(d)$. For simplicity's sake we will also use the simplified notation

$R_{\bar{x}}^{\bar{v}}$ for renomination composition. The basic composition operations of renominative logics are \vee , \neg , and $R_{\bar{x}}^{\bar{v}}$.

At the *quantifier* level, all basic (object) values can be used to construct different nominative sets to which quasiary predicates can be applied. This allows one to introduce the compositions of quantification $\exists x$ in style of Kleene's strong quantifiers. The basic compositions of logics of the quantifier level are \vee , \neg , $R_{\bar{x}}^{\bar{v}}$, and $\exists x$.

At the *quantifier-equational* level, new possibilities arise for equating and differentiating values using special 0-ary compositions, i.e., parametric equality predicates $=_{xy}$. Basic compositions of logics of the quantifier-equational level are \vee , \neg , $R_{\bar{x}}^{\bar{v}}$, $\exists x$, and $=_{xy}$.

All specified logics (renominative, quantifier, and quantifier-equational) are based on algebras which have only one sort: a class of quasiary predicates.

For quasiary predicate-function logics we identify function level and function-equational levels.

At the *function* level, we have extended capabilities of formation of new arguments for functions and predicates. In this case it is possible to introduce the superposition composition $S^{\bar{x}}$ (see [6, 10]), which formalizes substitution of functions into predicate. It also seems natural to introduce special 0-ary compositions, called denaming functions $'x$. Given a nominative set, $'x$ yields a value of the name x in this set. Introduction of such functions allows one to model renomination compositions with the help of superposition. The basic compositions of logics of the function level are \vee , \neg , $S^{\bar{x}}$, $\exists x$, and $'x$.

At the function-equational level a special equality composition $=$ can be introduced additionally [10]. The basic compositions of logics of the function-equational level are \vee , \neg , $S^{\bar{x}}$, $\exists x$, $'x$, and $=$. At this level different classes of first-order logics can be presented.

This means that two-sorted algebras (with sets of predicates and functions as sorts and above-mentioned compositions as operations) form a semantic base for first-order CNL.

The level of *program logics* is quite rich. First, program compositions should be defined that describe the structure of programs. In the simplest case these are:

1. assignment composition $AS^x: Fn^{V,A} \xrightarrow{t} Prg^{V,A}$,
2. composition of sequential execution $\bullet: Prg^{V,A} \times Prg^{V,A} \xrightarrow{t} Prg^{V,A}$,
3. conditional composition $IF: Pr^{V,A} \times Prg^{V,A} \times Prg^{V,A} \xrightarrow{t} Prg^{V,A}$,
4. cycling composition $WH: Pr^{V,A} \times Prg^{V,A} \xrightarrow{t} Prg^{V,A}$.

Then we should define compositions specifying program properties. Here we only mention a composition which formalizes the notion of assertion in *Floyd-Hoare logic*. From a semantic point of view an assertion scheme of the form $\{P\}prog\{Q\}$ may be considered as composition FH , which given two quasiary predicates P (precondition), Q (postcondition), and a bi-quasiary function (a program) $prog$ produces new quasiary predicate denoted by $FH(P, prog, Q)$. At this level we obtained a three-sorted predicate-function-program algebra. Classes of terms of this algebra may be considered as sets of formulas (or their components) of corresponding logics.

Having described classification of composition-nominative logics we can formulate a task of investigation of logics presented in this classification. For many of such logics axiomatic calculi were constructed and their properties were investigated [10, 12].

In this paper we will consider the satisfiability problem for logics of quantifier-equational level. This problem for logics of the previous levels (propositional, renominative, and quantifier) was considered in [13]. We choose a reduction method that reduces the satisfiability problem of composition-nominative logic to the satisfiability in classical logic. To simplify this reduction we will use an intermediate logic with unessential variables. Thus, we will define three logics of quantifier-equational level: composition-nominative logic, logic with unessential variables, and classical first-order logic.

3 Formal Definitions of Logics of Quantifier-Equational Level

At first, we describe a general mechanism of specifying composition-nominative logics and then provide definitions for the logics considered in this paper. To do this we should specify three logic components that reflect the semantic-syntactic scheme of logic definition:

- *semantic component*: a class of algebras of quasiary predicates that forms a semantic base for a logic. In our case we consider algebras of the form

$AQE(V, A) = \langle Pr^{V,A}, \vee, \neg, R_{\bar{x}}^{\bar{v}}, \exists x, =_{xy} \rangle$ for various sets of atomic values A (recall

that $Pr^{V,A} = {}^V A \xrightarrow{P} Bool$ is a class of partial predicates over ${}^V A$);

- *syntactic component*: a logic language specified by a class of logic formulas. This class is determined by the logic signature Σ , which includes the infinite set of names V , a set Ps of predicate symbols and a set Cs of composition symbols; the set of formulas $Fr(\Sigma)$ is constructed inductively over the set of atomic formulas $AFr(\Sigma)$ with the help of symbols of compositions;
- *interpretational (denotational) component*: a parametric total mapping that prescribes to a formula its meaning as a predicate. Parameters are algebra $AQE(V, A)$ and interpretation for atomic formulas $I: AFr(V, Ps) \xrightarrow{I} Pr^{V,A}$ called σ -interpretation. A pair $(AQE(V, A), I)$ is called a model of the logic. Given a model $M = (AQE(V, A), I)$ an interpretational mapping for each formula Φ specifies its meaning as a quasiary predicate in $AQE(V, A)$ denoted Φ_M . Usually models are represented in simplified form, say $J = (V, A, I)$, called π -interpretations; then the meaning of the formula is denoted Φ_J .

A logic defined according to this scheme is denoted $L(\Sigma)$.

3.1 Algebras of Quasiary Predicates of Quantifier-Equational Level

Semantic base of composition-nominative logics is specified by classes of data, predicates, and compositions. The latter are determined by the abstraction level of logic under consideration and are the same for all logics of the level. As was formulated earlier, for the logics of quantifier-equational level (QE-level) the class of compositions consists of basic propositional connectives, renomination composition, quantifiers, and equality predicate. The compositions (except propositional connectives) are parametric with parameters from an infinite set of names V .

Therefore we consider the following set of composition symbols:

$Cs_{QE}(V) = \{ \vee, \neg \} \cup \{ R_{\bar{x}}^{\bar{v}} \mid \bar{v} = (v_1, \dots, v_n), \bar{x} = (x_1, \dots, x_n), \bar{v} \text{ is a list of distinct names, } v_i, x_i \in V \text{ for all } i \in \{1, \dots, n\}, n \geq 0 \} \cup \{ \exists x \mid x \in V \} \cup \{ =_{xy} \mid x, y \in V \}$.

For the sake of simplicity we will write $Cs_{QE}(V) = \{ \vee, \neg, R_{\bar{x}}^{\bar{v}}, \exists x, =_{xy} \}$.

Given an algebra $AQE(V, A) = \langle Pr^{V,A}, \vee, \neg, R_{\bar{x}}^{\bar{v}}, \exists x, =_{xy} \rangle$ we now define interpretation of composition symbols. Again, for simplicity's sake we will use the same notations for compositions (as operations in the algebra) and their symbols.

In definitions of compositions we will use the following notation:

- $p(d) \downarrow$ means that a predicate p is defined on data d ;
- $p(d) \downarrow = b$ means that a predicate p is defined on data d with a Boolean value b ;
- $p(d) \uparrow$ means that a predicate p on d is undefined;
- for nominative data representation we use the form $d = [v_i \mapsto a_i \mid i \in I]$. *Nominative membership relation* is denoted by \in_n . Thus, $v_i \mapsto a_i \in_n d$ means that the value of v_i in d is defined and is equal to a_i ; this can be written in another form as $d(v_i) \downarrow = a_i$.

Propositional compositions are defined by the following formulas ($p, q \in Pr^{V,A}$, $d \in {}^V A$):

$$(p \vee q)(d) = \begin{cases} T, & \text{if } p(d) \downarrow = T \text{ or } q(d) \downarrow = T, \\ F, & \text{if } p(d) \downarrow = F \text{ and } q(d) \downarrow = F, \\ \text{undefined} & \text{in other cases.} \end{cases}$$

$$(\neg p)(d) = \begin{cases} T, & \text{if } p(d) \downarrow = F, \\ F, & \text{if } p(d) \downarrow = T, \\ \text{undefined} & \text{if } p(d) \uparrow. \end{cases}$$

Unary renomination composition $R_{\bar{x}}^{\bar{v}}$ is a mapping $R_{\bar{x}}^{\bar{v}}: Pr^{V,A} \xrightarrow{t} Pr^{V,A}$, where $\bar{v} = (v_1, \dots, v_n)$ and $\bar{x} = (x_1, \dots, x_n)$ are lists of names from a set V ; names from \bar{v} are called upper names of renomination composition and should be distinct, $n \geq 0$. Please note that $R_{\bar{x}}^{\bar{v}}$ is a parametric composition which represents a class of renomination compositions with different parameters, which are elements of V . This composition is defined by the following formula ($p \in Pr^{V,A}$, $d \in {}^V A$):

$$(R_{x_1, \dots, x_n}^{v_1, \dots, v_n} p)(d) = p([v \mapsto a \in_n d \mid v \notin \{v_1, \dots, v_n\}] \nabla [v_i \mapsto d(x_i) \mid d(x_i) \downarrow, i \in \{1, \dots, n\}]).$$

The ∇ operation is defined as follows: if d_1 and d_2 are two nominative sets, then $d = d_1 \nabla d_2$ consists of all named pairs of d_2 and only those pairs of d_1 , whose names are not defined (do not have values) in d_2 .

Unary parametric composition of existential quantification $\exists x$ with the parameter $x \in V$ is defined by the following formula ($p \in Pr^{V,A}$, $d \in {}^V A$):

$$(\exists x p)(d) = \begin{cases} T, & \text{if } b \in A \text{ exists : } p(d \nabla x \mapsto b) \downarrow = T, \\ F, & \text{if } p(d \nabla x \mapsto a) \downarrow = F \text{ for each } a \in A, \\ \text{undefined} & \text{in other cases.} \end{cases}$$

Here $d \nabla x \mapsto a$ is a shorter form for $d \nabla [x \mapsto a]$.

Finally, *null-ary parametric equality composition* $=_{xy}$ ($x, y \in V$) is defined as follows:

$$=_{xy}(d) = \begin{cases} T, & \text{if } d(x) \downarrow, d(y) \downarrow \text{ and } d(x) = d(y), \\ T, & \text{if } d(x) \uparrow \text{ and } d(y) \uparrow, \\ F & \text{otherwise} \end{cases}$$

Now we will give definitions for all logics with a fixed infinite set of names V and a fixed set of predicate symbols Ps . Note that according to the tradition elements of V are also called *variables*. As semantic components for all logics are the same, we need to define only syntactic and interpretational components.

3.2 Composition-Nominative Logic $L_{QE}(\Sigma_{QE})$ of Quantifier-Equational Level

1. *Syntactic component.* A tuple $\Sigma_{QE} = (V, \{\vee, \neg, R_{\bar{x}}^{\bar{v}}, \exists x, =_{xy}\}, Ps)$ is called a *signature* of composition-nominative logic of QE-level. Taking into consideration that a set of composition symbols is determined by the set of variables V , we will use for a signature a simplified notation (V, Ps) . Language of $L_{QE}(\Sigma_{QE})$ is represented by a class of formulas $Fr_{QE}(V, Ps)$, which is defined inductively:

- If $P \in Ps$ then $P \in Fr_{QE}(V, Ps)$. Such formulas are called atomic and belong to the class $AFr_{QE}(V, Ps)$ of atomic formulas.
- If $x, y \in V$ then $=_{xy} \in Fr_{QE}(V, Ps)$. Such formulas are called atomic and belong to the class $AFr_{QE}(V, Ps)$ of atomic formulas.
- If $\Phi, \Psi \in Fr_{QE}(V, Ps)$ then $(\Phi \vee \Psi) \in Fr_{QE}(V, Ps)$ and $\neg \Phi \in Fr_{QE}(V, Ps)$.
- If $\bar{v} = (v_1, \dots, v_n)$, $\bar{x} = (x_1, \dots, x_n)$, \bar{v} is a list of distinct variables, $v_i, x_i \in V$ for all $i \in \{1, \dots, n\}$, $n \geq 0$, $\Phi \in Fr_{QE}(V, Ps)$ then $R_{\bar{x}}^{\bar{v}} \Phi \in Fr_{QE}(V, Ps)$.
- If $x \in V$, $\Phi \in Fr_{QE}(V, Ps)$ then $\exists x \Phi \in Fr_{QE}(V, Ps)$.

Note, that predicate symbols and symbols of null-ary compositions are atomic formulas.

2. *Interpretational component.* Let $AQE(V, A) = \langle Pr^{V,A}, \vee, \neg, R_{\bar{x}}^{\bar{v}}, \exists x, =_{xy} \rangle$ be an algebra of quasiary predicates of quantifier-equational level. In this algebra composition symbols obtain their interpretations as operations over predicates. In particular, atomic formulas for null-ary compositions $=_{xy}$ are interpreted as equality predicates in this algebra. Thus, we need to specify interpretation mappings for predicate symbols only.

This is done with a mapping $I_{QE}^{Ps}: Ps \xrightarrow{t} Pr^{V,A}$ called a σ -interpretation. Having the interpretational mapping for predicate symbols, we can compositionally construct interpretational mapping for all formulas. A pair $(AQE(V, A), I_{QE}^{Ps})$ is called a model for $L_{QE}(\Sigma_{QE})$. A model is determined by a tuple $J_{QE}^{Ps} = (V, A, I_{QE}^{Ps})$ called π -interpretation. In simplified form interpretations will be denoted J . For interpretation J and a formula Φ the meaning of Φ is denoted Φ_J .

3.3 Composition-Nominative Logic $L_{QEU}(\Sigma_{QEU})$ of Quantifier-Equational Level with Unessential Variables

Unessential variables play a role of additional memory and are used for “storing” values during formula transformations. We assume that a set U of unessential variables is an infinite subset of V ($U \subseteq V$). Informally speaking, logic with unessential variables is a logic $L_{QE}(\Sigma_{QE})$ with restriction on interpretations of predicate symbols specified by the set U .

1. *Syntactic component.* A tuple $\Sigma_{QEU} = (V, U, \{\vee, \neg, R_{\bar{x}}^{\bar{v}}, \exists x, =_{xy}\}, Ps)$ is called a signature of CNL of QE-level with unessential variables. A class of formulas for L_{QEU} is $Fr_{QEU}(V, U, Ps) = Fr_{QE}(V, Ps)$.

2. *Interpretational component.* Let $AQE(V, A) = \langle Pr^{V,A}, \vee, \neg, R_{\bar{x}}, \exists x, =_{xy} \rangle$ be an algebra of quasiary predicates of quantifier-equational level. By calling variables from U unessential we actually put a restriction on interpretations of predicate symbols. This restriction asserts that in σ -interpretation $I_{QEU}^{Ps} : Ps \xrightarrow{t} Pr^{V,A}$ for every $P \in Ps$ and for every $d \in {}^V A$ the value of $I_{QEU}^{Ps}(P)(d)$ does not depend on values of variables from the set U in d . Formally, for every $d \in {}^V A$ the values $I_{QEU}^{Ps}(P)(d)$ and $I_{QEU}^{Ps}(P)(d \parallel U)$ should either be equal or be undefined simultaneously. Here $d \parallel U = \{v \mapsto a \in {}_n d \mid v \notin U\}$. A π -interpretation will be denoted $J_{QEU}^{Ps} = (V, U, A, I_{QEU}^{Ps})$. Indexes may be omitted if they are clear from the context.

This completes a formal definition of logic $L_{QEU}(\Sigma_{QEU})$.

3.4 Classical First-Order Predicate Logic $L_{QECL}(\Sigma_{QECL})$ with Equality

A definition of classical logic differs from definitions of CNL because it is oriented not on quasiary but on n -ary predicates.

1. *Syntactic component.* A tuple $\Sigma_{QECL} = (V, \{\vee, \neg, \exists x, =\}, Ps, arity)$ is called a signature of a classical logic with equality (here $arity: Ps \xrightarrow{t} \{0,1,2, \dots\}$ is a function that for each predicate symbol yields its arity). A signature in a simplified form is denoted $(V, Ps, arity)$. The language $Fr_{QECL}(V, Ps, arity)$ is defined inductively:

- If $P \in Ps$, $arity(P)=n$, and $x_1, \dots, x_n \in V$, then $P(x_1, \dots, x_n) \in Fr_{QECL}(V, Ps, arity)$. Such formulas are called atomic and belong to the class $AFr_{QECL}(V, Ps, arity)$ of atomic formulas.
- If $x, y \in V$ then $x=y \in Fr_{QECL}(V, Ps, arity)$. Such formulas are called atomic and belong to the class $AFr_{QECL}(V, Ps, arity)$ of atomic formulas.
- If $\Phi, \Psi \in Fr_{QECL}(V, Ps, arity)$ then $(\Phi \vee \Psi) \in Fr_{QECL}(V, Ps, arity)$ and $\neg \Phi \in Fr_{QECL}(V, Ps, arity)$.
- If $x \in V, \Phi \in Fr_{QECL}(V, Ps, arity)$ then $\exists x \Phi \in Fr_{QECL}(V, Ps, arity)$.

2. *Interpretational component.* Let $AQE(V, A) = \langle Pr^{V,A}, \vee, \neg, R_{\bar{x}}, \exists x, =_{xy} \rangle$ be an algebra of quasiary predicates of QE-level (for classical logic we assume that A is non-empty). Note that the renomination composition is present as operation in this algebra, though it is not explicitly used in classical logic. Formulas of the language are interpreted as predicates in this algebra. Atomic formula $x=y$ is interpreted as a predicate $=_{xy}$. To give an interpretation of atomic formulas of the form $P(x_1, \dots, x_n)$ we need to specify an interpretational mapping for predicate symbols. In case of classical logic it is specified by a mapping $I_{NAr}^{Ps} : Ps \xrightarrow{t} \bigcup_{n \geq 0} (A^n \xrightarrow{t} Bool)$ such that

$I_{NAr}^{Ps}(P) \in A^n \xrightarrow{t} Bool$ if $arity(P) = n$ for $P \in Ps$. This mapping interprets predicate symbols as total n -ary predicates. Thus, π -interpretations have the form $J_{CLE}^{Ps} = (V, A, arity, I_{NAr}^{Ps})$. Such π -interpretation J_{CLE}^{Ps} (or simply J) for every

atomic formula $P(x_1, \dots, x_n)$ defines its meaning in Pr^{VA} as a predicate $P(x_1, \dots, x_n)_J$ such that $P(x_1, \dots, x_n)_J(d) = I_{NAr}^{Ps}(P)(d(x_1), \dots, d(x_n))$ for every $d \in {}^VA$; if one of the values $d(x_1), \dots, d(x_n)$ is not defined then $P(x_1, \dots, x_n)_J$ is undefined on d . Let us note that in classical logic d is called *variable valuation* or *variable assignment*. The meaning Φ_J of a complex formula $\Phi \in Fr_{QECL}(Ps, V, arity)$ is defined in a usual way.

For all three logics derived compositions (such as conjunction $\&$, universal quantification $\forall x$, negated equality \neq_{xy} etc.) are defined in a traditional way. In the sequel we consider formulas in their traditional form using infix operations and brackets; brackets can be omitted according to common rules for the priorities of operations (priority of the binary disjunction is weaker than priority of unary operations). We will also consider a more general case for I_{NAr}^{Ps} permitting *partial* n -ary predicates as values of predicate symbols, thus, $I_{NAr}^{Ps}(P) \in A^n \xrightarrow{P} Bool$; still, this generalization does not affect the satisfiability problem due to monotonicity of considered compositions under predicate extensions [13].

To simplify notation we will often omit parameters of logic signatures and write simply L_{QE} , L_{QEU} , and L_{QECL} ; for classes of formulas we use notations Fr_{QE} , Fr_{QEU} , and Fr_{QECL} ; formulas of these classes will be called QE-, QEU-, and CL-formulas, π -interpretations in L_{QE} , L_{QEU} , L_{QECL} will also be called QE-, QEU-, CL-interpretations respectively.

3.5 Satisfiability Problem

For all three logics the definition of satisfiability can be given in the same way.

A formula Φ is called *satisfiable in a π -interpretation J* if there is $d \in {}^VA$ such that $\Phi_J(d) \downarrow = T$. We shall denote this by $J \approx \Phi$. A formula Φ is called *satisfiable* if there exists an interpretation J in which Φ is satisfiable. We shall denote this as $\approx \Phi$. We call formulas Φ and Ψ *equisatisfiable* if they are either both satisfiable or both not satisfiable (i.e., unsatisfiable). When needed we will underline the corresponding logic in the satisfiability sign \approx , e.g. \approx_{QE} , \approx_{QEU} , or \approx_{QECL} .

Satisfiability of a formula is related to its validity. A formula Φ is called *valid in a π -interpretation J* if there is no $d \in {}^VA$ such that $\Phi_J(d) \downarrow = F$. We shall denote this as $J \models \Phi$, which means that Φ is not refutable in J . A formula Φ is called *valid* if $J \models \Phi$ for every interpretation J . We call formulas Φ and Ψ *equivalent* if $\Phi_J = \Psi_J$ for every interpretation J .

Due to possible presence of a nowhere defined predicate (which is a valid predicate) we do not have in CNL the property that Φ is satisfiable if Φ is valid (which holds for classical first-order logic). But reduction of satisfiability to validity still holds in CNL: formula Φ is satisfiable in a π -interpretation J iff $\neg\Phi$ is not valid in J .

4 Reduction of Satisfiability Problem for $L_{QE}(\Sigma_{QE})$

The problem discussed in this paper is to check whether $\models_{QE} \Phi$ holds given an arbitrary formula $\Phi \in Fr_{QE}(W, Ps)$; here we choose W as an initial set of variables in the considered logic. Our main aim is to transform this QE-formula Φ to an equisatisfiable formula Φ_{CL} of classical first-order predicate logic with equality so that we can use existent methods for solving this problem developed for classical logic. To carry out necessary equivalent transformations we need to consider Φ in an intermediate logic – CNL of QE-level with unessential variables – extending the initial set of variables W with a set U of unessential variables ($W \cap U = \emptyset$). For these needs we will consider a logic L_{QEU} with the signature $\Sigma_{QEU} = (V, U, \{\vee, \neg, R_{\bar{x}}^{\bar{v}}, \exists x, =_{xy}\}, Ps)$, where $V = W \cup U$. Within L_{QEU} we transform Φ to a formula Φ_{UR} being in a special normal form; then the latter formula is translated to its classical counterpart Φ_{CL} .

The overall circular reduction scheme is grounded on following statements.

1. From $\models_{QE} \Phi$ follows $\models_{QEU} \Phi$ (lemma 1).
2. From $\models_{QEU} \Phi$ follows $\models_{QEU} \Phi_{UR}$ (lemma 2, 3).
3. From $\models_{QEU} \Phi_{UR}$ follows $\models_{QECL} \Phi_{CL}$ (lemma 4).
4. From $\models_{QECL} \Phi_{CL}$ follows $\models_{QEU} \Phi_{UR}$ (lemma 5).
5. From $\models_{QEU} \Phi_{UR}$ follows $\models_{QEU} \Phi$ (lemma 2).
6. From $\models_{QEU} \Phi$ follows $\models_{QE} \Phi$ (lemma 6).

Lemma 1. Let $\Phi \in Fr_{QE}(W, Ps)$. Then from $\models_{QE} \Phi$ follows $\models_{QEU} \Phi$.

Consider the transformation rules (T1-T9) of the form $\Phi_l \mapsto \Phi_r$, where $\Phi_l, \Phi_r \in Fr_{QEU}(V, U)$.

$$T1) R_{\bar{x}}^{\bar{v}} =_{xy} \mapsto =_{\bar{x}\bar{y}}$$

$$T2) R_{\bar{x}}^{\bar{v}} \neg \Phi \mapsto \neg R_{\bar{x}}^{\bar{v}} \Phi$$

$$T3) R_{\bar{x}}^{\bar{v}} (\Phi_1 \vee \Phi_2) \mapsto R_{\bar{x}}^{\bar{v}} \Phi_1 \vee R_{\bar{x}}^{\bar{v}} \Phi_2$$

$$T4) R_{x_1, \dots, x_n, y_1, \dots, y_m}^{v_1, \dots, v_n, w_1, \dots, w_m} R_{s_1, \dots, s_n, z_1, \dots, z_k}^{v_1, \dots, v_n, u_1, \dots, u_k} \Phi \mapsto R_{\alpha_1, \dots, \alpha_n, \gamma_1, \dots, \gamma_m, \beta_1, \dots, \beta_k}^{v_1, \dots, v_n, w_1, \dots, w_m, u_1, \dots, u_k} \Phi$$

$$T5) R_{\bar{x}}^{\bar{v}} \exists y \Phi \mapsto \exists y R_{\bar{x}}^{\bar{v}} \Phi, \text{ when } y \notin \{\bar{v}, \bar{x}\}$$

$$T6) R_{z, \bar{x}}^{y, \bar{v}} \exists y \Phi \mapsto \exists y R_{\bar{x}}^{\bar{v}} (\Phi)$$

T7) $R_{y, \bar{x}}^{z, \bar{v}} \exists y \Phi \mapsto \exists u R_{y, \bar{x}}^{z, \bar{v}} R_u^y \Phi$, $u \in U$, u does not occur in the formula on the left hand side of the rule.

T8) $R_{\bar{q}}^{\bar{u}} P \mapsto R_{z, \bar{q}}^{z, \bar{u}} P$ (in case when vectors \bar{u}, \bar{v} are empty this rule is represented as $P \mapsto R_z^z P$).

$$\text{T9) } R_{q_1, \dots, q_i, \dots, q_j, \dots, q_n}^{u_1, \dots, u_i, \dots, u_j, \dots, u_n} P = R_{q_1, \dots, q_j, \dots, q_i, \dots, q_n}^{u_1, \dots, u_j, \dots, u_i, \dots, u_n} P$$

Here for the rule T1 $\tilde{x} = x(\bar{v}/\bar{x})$, $\tilde{y} = y(\bar{v}/\bar{x})$, for the rule T4 $\alpha_i = s_i(v_1, \dots, v_n, w_1, \dots, w_m / x_1, \dots, x_n, y_1, \dots, y_m)$, $\beta_j = z_j(v_1, \dots, v_n, w_1, \dots, w_m / x_1, \dots, x_n, y_1, \dots, y_m)$, where $r(b_1, \dots, b_q / c_1, \dots, c_q) = r$ if $r \notin \{b_1, \dots, b_q\}$, $r(b_1, \dots, b_q / c_1, \dots, c_q) = c_i$ if $r = b_i$ for some i .

The rule T4 represents explicitly the result of functional composition of parameters of two successive renominations.

The rule T7 permits to assume w.l.o.g. that all quantified variables in initial formula are different.

Lemma 2. Let $\Phi_l, \Phi_r \in Fr_{QEU}(V, U, Ps)$ be such formulas that Φ_r is a result of application of some T1-T9 rule to Φ_l . Then Φ_l and Φ_r are equisatisfiable in L_{QEU} .

A formula Φ is said to be in *unified renominative normal form* (URNF) if the following requirements are satisfied:

- the renomination composition is only applied in Φ to predicate symbols. It means that for every sub-formula of the form $R_{\bar{x}}^{\bar{v}} \Psi$ we have that $\Psi \in Ps$;
- for every pair of its renominative atoms $R_{\bar{q}}^{\bar{u}} P$ and $R_{\bar{y}}^{\bar{w}} Q$ we have that vectors \bar{u} and \bar{w} coincide; so, in all renominative atoms the lists of their upper names are the same;
- for every renominative atom $R_{\bar{x}}^{\bar{v}} P$ and every quantifier $\exists y$ that occurs in the initial formula Φ we have that $y \in \bar{v}$.

When formula is in *URNF* we call its atomic subformula $R_{\bar{x}}^{\bar{v}} P$ a *renominative atom* ($P \in Ps$). Note that if a formula is in *URNF* then every its subformula is in *URNF* as well.

Lemma 3. Given an arbitrary formula $\Phi \in Fr_{QEU}(V, U, Ps)$ we can construct its unified renominative normal form $urnf[\Phi]$ by applying rules T1-T9.

According to lemmas 2 and 3, we can think of a total multi-valued (non-deterministic) mapping $urnf : Fr_{QEU} \xrightarrow{im} Fr_{QEU}$ that transforms in a satisfiability-preserving way every QEU-formula to its URNF.

In order to reduce the satisfiability problem in L_{QEU} to that of L_{QECL} we extend the set of basic values A with additional value ε . Informally, this value will represent undefined components of nominative sets.

We formalize the syntactical reduction $clf : Fr_{QEU}(V, U, Ps) \xrightarrow{t} Fr_{QECL}(V, Ps, arity)$ of QEU-formulas in unified renominative normal form to CL-formulas inductively as follows:

1. $clf[P] = P$
2. $clf[=_{xy}] = =_{xy}$
3. $clf[R_{x_1, \dots, x_n}^{v_1, \dots, v_n} P] = P(x_1, \dots, x_n)$
4. $clf[\neg \Phi] = \neg clf[\Phi]$

5. $clf[(\Phi_1 \vee \Phi_2)] = (clf[\Phi_1] \vee clf[\Phi_2])$
6. $clf[\exists x\Phi] = \exists x(x \neq e \ \& \ clf[\Phi])$, $e \in U$, e is a predefined variable.

Note that all applications of the 6-th rule introduce the same variable e ; e is some predefined variable from U in the sense that it does not occur in URNF.

This reduction transforms the formula to the language of classical logic but preserves its satisfiability.

Given a formula $\Phi \in Fr_{QEU}(V, U, Ps)$ in unified renominative normal form we denote by $V_\Phi \subseteq V$ the set of all variables that occur as upper names in renominative atoms of Φ .

Lemma 4. Let Φ be a formula in unified renominative normal form, $\Phi \in Fr_{QEU}(V, U, Ps)$. Then from $\models_{QEU} \Phi$ follows $\models_{QECL} clf[\Phi]$.

Lemma 5. Let Φ be a formula in renominative normal form, $\Phi \in Fr_{QEU}(V, U, Ps)$. Then from $\models_{QECL} clf[\Phi]$ follows $\models_{QEU} \Phi$.

Lemma 6. Let $\Phi \in Fr_{QE}(W, Ps)$. Then from $\models_{QEU} \Phi$ follows $\models_{QE} \Phi$.

Lemmas 1-6 justify all reductions described in the article and the main theorem of the article.

Theorem. Let $\Phi \in Fr_{QE}(W, Ps)$. Then $\models_{QE} \Phi$ if and only if $\models_{QECL} urnf[clf[\Phi]]$.

The theorem states the reduction of satisfiability problem in composition-nominative logic of quantifier-equational level to the satisfiability problem in classical first-order logic with equality.

Let us illustrate the method proposed on a simple example.

Example. Consider the following QE-formula Φ with one predicate symbol P :

$$\Phi = P \ \& \ R_x^z (=_{xy} \ \& \ \forall z \neg P)$$

Let us construct its unified renominative normal form Φ_{UR} .

$$\Phi = P \ \& \ R_x^z (=_{xy} \ \& \ \forall z \neg P) \mapsto / \text{push the renomination down to predicate symbols/}$$

$$\mapsto P \ \& \ =_{xy} \ \& \ R_x^z (\forall z \neg P) \mapsto / \text{renomination is removed due to T6/} \mapsto$$

$$\mapsto P \ \& \ =_{xy} \ \& \ (\forall z \neg P) \mapsto / \text{add } R_z^z \text{ to } P \text{ as the predicate occurs under } \forall z / \mapsto$$

$$\mapsto P \ \& \ =_{xy} \ \& \ (\forall z \neg R_z^z P) \mapsto / \text{unify renominative atoms} / \mapsto$$

$$\mapsto R_z^z P \ \& \ =_{xy} \ \& \ \forall z \neg R_z^z P = \Phi_{UR}.$$

Note that we use derived transformation rules that handle compositions $\&$ and \forall .

Now $\Phi_{CL} = cnl[\Phi_{UR}] = P(z) \ \& \ x = y \ \& \ \forall z((z \neq e) \rightarrow \neg P(z))$. Formula Φ_{CL} is satisfiable in L_{CL} . That means that Φ is satisfiable in L_{QE} .

Indeed, let $J = (W, A, I)$ be such an interpretation that $W = \{x, y, z\}$, $A = \{1, 2\}$. Let $I(P)(d) \downarrow = F$ if a pair $z \mapsto a \in_n d$ for some $a \in A$ and T in all other cases. In other words, the predicate P takes the value T on some data d if the variable z is undefined in d . Now we have that $\Phi_J([x \mapsto 1, y \mapsto 1]) \downarrow = T$.

5 Related work

Many different aspects of the composition-nominative approach such as partiality, compositionality, nominativity, have long history of development, which is also reflected in works in the field of logic and computer science.

The importance of partiality, for example, was already being discussed in detail by the time of 80-ties [14], and many different approaches have emerged since that time. In [15, 16] there is a survey of some of those and a comparison of different formalisms. Partiality receives more and more attention nowadays, the support for partial functions is being introduced in theorem proving systems and validity checkers [17, 18].

Compositionality can be traced back to works of G. Frege; the history of this principle is presented in [19]. The importance of the compositionality principle grows due to the necessity of investigation and verification of complex systems [20, 21], in particular, concurrent systems [22]. Our approach takes compositionality as a basic principle, thus, the constructed formal languages are compositional by construction when we consider functions (predicates) as meanings of expressions (of formulas).

Nominativity is also a fundamental aspect not only in computer science but in other branches of science as well, especially in philosophy. This topic requires a special treatment, but here we would like to mention nominal logic [23] only, which has similarities with the logic defined in this paper. Nominal logic addresses such special questions of nominativity as name bindings, swapping, and freshness. The predicates investigated in nominal logic should be equivariant (their validity is invariant under name swapping); in our work we consider general classes of partial predicates.

A thorough comparison of composition-nominative approach with other approaches that address compositionality, nominativity or allow reasoning about partial functions and predicates is by far beyond the scope of this paper, but still we would like to stress on the important differences. Our approach is based on algebras of partial predicates over nominative data, and especially, algebras of quasiary functions and predicates as opposed to traditional algebras of n -ary functions and predicates. It involves new compositions, in particular, renomination composition, which take into account nominative aspects of data structures. Composition-nominative approach also prescribes the semantic-syntactic style of logic definitions. This style simplifies construction and investigation of such logics.

6 Conclusions

This paper investigates the satisfiability problem for composition-nominative logic (CNL) of quantifier-equational level. As a main result we have shown that this problem can be reduced by using more powerful language to the satisfiability problem for classical predicate logic with equality. Thus, existent state-of-the-art methods and techniques for checking satisfiability in classical logics can also be applied to CNL.

Future work on the topic will include investigation of satisfiability problem for richer CNL of predicate-function level and for CNL over hierarchic nominative data. Hierarchic data permit to represent such complex structures as lists, stacks, arrays etc; thus, such logics will be closer to program models with more rich data types. Another

direction is related with identification of classes of formulas in various types of CNL for which satisfiability problem can be solved efficiently. In particular, this concerns specialized theories, where some predicates have specific interpretations and several axioms shall hold for such interpretations. This is often referred to as satisfiability modulo theory (SMT) problem [24]. At last, prototypes of software systems for satisfiability checking in CNL should be developed.

References

1. Mendelson E.: *Introduction to Mathematical Logic*, 4th ed. Chapman & Hall, London (1997)
2. Kroening D., Strichman O.: *Decision Procedures – an Algorithmic Point of View*. Springer-Verlag, Berlin Heidelberg (2008)
3. Marques-Silva J.: *Practical Applications of Boolean Satisfiability*. In: *Workshop on Discrete Event Systems (28-30 May 2008, Goteborg, Sweden)*, pp. 74--80 (2008)
4. Nieuwenhuis R., Oliveras A., Tinelli C.: *Solving SAT and SAT modulo theories: from an abstract Davis-Putnam-Logemann-Loveland procedure to DPLL(T)*. *J ACM* 53, pp. 937--977 (2006)
5. de Moura L., Bjørner N.: *Satisfiability Modulo Theories: Introduction and Applications*. *COMMUN ACM* 54 (9), pp. 69--77 (2011)
6. Nikitchenko, N.S.: *A Composition Nominative Approach to Program Semantics*. Technical Report IT-TR 1998-020, Technical University of Denmark (1998)
7. Basarab I.A., Gubsky B.V., Nikitchenko N.S., Red'ko V.N.: *Composition Models of Databases*. In: Eder J., Kalinichenko L.A. (eds.), *East-West Database Workshop (Workshops in Computing Series)*, pp. 221--231. Springer, London (1995)
8. Nielson H.R., Nielson F.: *Semantics with Applications: A Formal Introduction*. John Wiley & Sons Inc (1992)
9. Nikitchenko M.S.: *Composition-nominative aspects of address programming*. *Kibernetika I Sistemnyi Analiz* 6, pp. 24--35 (In Russian) (2009)
10. Nikitchenko M.S., Shkilnyak S.S.: *Mathematical logic and theory of algorithms*. Publishing house of Taras Shevchenko National University of Kyiv, Kyiv, (in Ukrainian) (2008)
11. Kleene, S. C.: *Introduction to Metamathematics*. Van Nostrand, New York (1952)
12. Shkilniak S. S.: *First-order logics of quasiary predicates*. *Kibernetika I Sistemnyi Analiz* 6, pp. 32--50, (in Russian) (2010)
13. Nikitchenko M.S., Tymofieiev V.G.: *Satisfiability Problem in Composition-Nominative Logics*. In: *Proceedings of the Eleventh International Conference on Informatics INFORMATICS'2011, Roznava, Slovakia, November 16-18*, pp. 75--80 (2011)
14. Blamey, S.: *Partial Logic*. In Gabbay D., Guenther F. (eds.), *Handbook of Philosophical Logic, Volume III*, D. Reidel Publishing Company (1986)
15. Jones C. B.: *Reasoning About Partial Functions in the Formal Development of Programs*. *ENTCS* 145, pp. 3--25 (2006)
16. Owe O.: *Partial Logics Reconsidered: A Conservative Approach*. *FORM ASP COMPUT* 5, pp. 208--223 (1997)
17. Mehta F. A.: *Practical Approach to Partiality A Proof Based Approach*. LNCS, vol. 5256, pp. 238--257. Springer, Heidelberg (2005)
18. Berezin S., Barrett C., Shikhanian I., Chechik M., Gurfinkel A., Dill D.L.: *A Practical Approach to Partial Functions in CVC Lite*. *ENTCS* 125, pp. 13--23 (2005)

19. Janssen T.M.V.: Compositionality. In van Benthem J., ter Meulen A. (eds.), *Handbook of Logic and Language*, pp. 417--473. Elsevier and MIT Press (1997)
20. de Roeper, W.-P., Langmaack H., Pnueli A. (eds.): *Compositionality: The Significant Difference*. LNCS, vol. 1536, VIII. Springer, Heidelberg (1998)
21. Bjørner D., Eir A.: *Compositionality: Ontology and Mereology of Domains*. LNCS, vol. 5930, pp. 22--59. Springer, Heidelberg (2010)
22. Dams D., Hannemann U., Steffen M. (eds.): *Concurrency, Compositionality, and Correctness, Essays in Honor of Willem-Paul de Roeper*. LNCS, vol. 5930. Springer, Heidelberg (2010)
23. Pitts, A. M.: *Nominal Logic, A First Order Theory of Names and Binding*. *INFORM COMPUT* 186, pp. 165--193 (2003)
24. Barrett C., Sebastiani R., Seshia S. A., Tinelli C.: *Satisfiability Modulo Theories*. In: Biere A., Heule M., van Maaren H., Walsh T. (eds.), *Handbook of Satisfiability*. IOS Press (2009)

Efficient Algorithm for Reachability Checking in Modeling

Alexander Letichevsky¹, Olexander Letychevskiy¹, and Vladimir Peschanenko²

¹ Glushkov Institute of Cybernetics of NAS of Ukraine, 40 Glushkova ave., Kyiv, Ukraine
let@cyfra.net, lit@iss.org.ua

² Kherson State University, 27, 40 Rokiv Zhovtnya str, Kherson, Ukraine
vladimirius@gmail.com

Abstract. The problem of reachability of the states of transition systems is considered hereby. The notions of partial unfolding and permutability of two operators (including the notion of statically permutable operators) are presented. New algorithm for reachability problem in terms of insertion modeling is described. An example of application of such algorithm is considered.

Keywords: insertion modeling, reachability, verification, interleaving, introduction

Key Terms: MathematicalModel, Process, Research.

1 Introduction

The verification of models of multiagent distributed systems and models of parallel computation usually need symbolic modeling and high level of abstraction. These models are highly nondeterministic because of symbolic nature of models and use of parallel composition adds a new level of non-determinism by interleaving. The main problem of verification is the problem of combinatorial explosion of states of the model. A state of model checking includes a lot of attributes and processes. The total number of states could be very large even if the number of processes is finite and all of the attributes are taken by finite number of values. Main source of combinatorial explosion of states are the number of processes and interleaving between them, nondeterministic behavior of them etc. Usually the systems are parallel and the number of their states grows exponentially with the number of processes. Our experience of verification of industrial systems shows that the total number of states is more than 21000. Obviously, model checking by naive enumeration of states is not feasible. Devoted to solving the problem many different technologies: developed methods for the partial order to reduce interleaving, used methods for determining the symmetry when verifying the equivalence of states, studied the information dependence to phase of verification components, applied techniques of abstraction, factorization, approximation, symbolic modeling etc. The standard model checking algorithms work only when the set of states reachable from the given initial state is

finite. Insertion modeling is a symbolic modeling with infinite number of states. There are various model checking techniques for infinite-state systems, but they are less developed than finite-state techniques and tend to place stronger limitations on the kind of systems and/or the properties that can be model checked. One of such techniques is presented in [1].

In Petri net theory there is well known the McMillan algorithm of unfolding that in many cases helps the exponential decreasing of interleaving in system analyses [2]. The book [3] generalizes this technique to the finite automata networks. So, the paper presents the new algorithm for reachability problem in terms of insertion modeling [4,5,6] for models with infinite number of states. The algorithm combines the ideas of economic unfolding of McMillan with on-line reachability checking using some general assumptions about the nature of information dependencies in the states of distributed system expressed in terms of permutability of actions.

So, the paper is devoted to the solution of a problem of reduce interleaving in insertion models with infinite number of states.

The algebra of behavior is presented in section Behavior Algebras, the verification environments and corresponded insertion function, predicate transformer are considered in section Verification Environments. The normal form of behavior is defined in section Behaviors Over Basis B . The problem of reachability of the states is described in section Verification. The notion of partial unfolding is examined in section Partial Unfolding. The optimization of partial unfolding by statically permutable operators is reviewed in section Static Permutability Property. The algorithm for reducing of interleaving for transitional systems is presented in section Algorithm of Reducing Interleaving. The example which demonstrates a good result of using the partial unfolding algorithm is presented in section Example of Application.

2 Behavior Algebras

Behavior algebra [5] is a kind of process algebra and it is used to express the behavior of agents (transition systems) considered up to bisimilarity or trace equivalence. To make economic unfolding we need to distinguish sequential and parallel behaviors. So we consider the following modification of the notion of behavior algebra. It is a multisorted algebra with three components: the algebra of *actions*, the algebra of *sequential behaviors*, and the algebra of *parallel behaviors*.

The algebra of sequential behaviors has operations of prefixing: $\langle action \rangle .$ $\langle sequential\ behavior \rangle$ and one internal operation of nondeterministic choice $(\langle \rangle + \langle \rangle)$ which is associative commutative and idempotent operation with neutral element 0 . We also consider the constant behavior Δ (successful termination) which is the common element of the algebra of sequential and the algebra of parallel behaviors. The operations of action algebra will be considered later.

The algebra of parallel behaviors has the parallel composition $\langle \rangle || \langle \rangle$ of sequential behaviors as the main binary operation. It is associative commutative (not idempotent) with the neutral element Δ . It also has the prefixing operation and nondeterministic choice. The algebra of sequential behaviors is implicitly included to

the algebra of parallel behaviors by identity $u = u \parallel \Delta$ (parallel composition with one component). The unfolding of parallel composition by interleaving will be considered only after inserting the agents formed by parallel composition into the environment.

3 Verification Environments

These environments $E = E(U, P, B)$ are defined by the following parameters: the set of *conditional expressions* U , the set of *operators* P , and the set of *basic behaviors* B . The set of conditions and the set of operators are used to define actions (it is a union of these two sets). The set of basic behaviors are used to define the behaviors of agents inserted into environment in the way which will be explained later. We also suppose that some logic language (first order or temporal) called as basic language is fixed to define the states of environment and checking conditions for verification. The conditional expressions also belong to this language.

The state of environment is represented as $E[u]$, where E is a statement of basic language and u is a parallel composition of sequential behaviors of agents inserted into environment. We suppose that operators are divided into the set of conditional and unconditional operators. Conditional operator has the form $\alpha \rightarrow a$ where α is a condition and a is an unconditional operator. Unconditional operator a is identified with conditional operator $1 \rightarrow a$. The associative product $()^*()$ and the function $pt: U \times P \rightarrow U$ (predicate transformer) are defined by the set of actions so that the following identities are valid:

$$\begin{aligned} pt(\alpha, \beta \rightarrow a) &= pt(\alpha \wedge \beta \rightarrow a) \\ pt(pt(\alpha, a), b) &= pt(\alpha, a^*b) \\ (\alpha \rightarrow a)^*(\beta \rightarrow b) &= pt(pt(\alpha, a) \wedge \beta, b) \\ \alpha^* \beta &= \alpha \wedge \beta \end{aligned}$$

Here α and β are conditions, a and b are unconditional operators.

Predicate transformer is supposed to be monotonic: $\alpha \rightarrow \beta \Rightarrow pt(\alpha, a) \rightarrow pt(\beta, a)$. In general case, the pt function is defined by some concrete syntax. One of the examples of such pair (*syntax*, pt) could be found [7].

Example. Basic language is a first order language. Conditions are formulae over simple attributes - the symbols that change their values when system changes its state. Formally they are considered as functional symbols with arity 0. Unconditional operators are assignments (parallel assignments, sequences of assignments and if-then-else operators, loops with finite number of repetitions etc.). As usually in this case

$$pt(\alpha(x), (x_1 := t_1(x), x_2 := t_2(x), \dots)) = \exists z(\alpha(z) \wedge (x_1 = t_1(z) \wedge x_2 = t_2(z) \wedge \dots))$$

Actually this is the strongest postcondition for precondition α .

Insertion function is defined by the following identities and rules.

1. $E[u, v] = E[u \parallel v]$, u, v are agents with sequential behavior (see sec. 1).

Identities for conditions.

2. $E[\alpha.u + v] = E[v]$, if $(E \wedge \alpha) = 0$.

3. $E[\alpha.\beta.u + v] = E[\alpha \wedge \beta.u + v]$, if $(E \wedge \alpha) \neq 0$ (merging conditions).

4. $E[\alpha.\beta \rightarrow a.u + v] = E[\alpha \wedge \beta \rightarrow a.u + v]$, if $(E \wedge \alpha \wedge \beta) \neq 0$. Special cases of these identities are obtained when $v=0$ or $\beta = 1$ as special cases.

5. $E[\alpha.\varepsilon] = E \wedge \alpha[\varepsilon]$, if $(E \wedge \alpha) = 0$.

Identities for operators.

6. $E[a.u + v] = E[v]$, if $pt(E, a) = 0$.

7. $E[a.u] = a.pt(E, a)[u \parallel \varphi(a, E)]$, if $pt(E, a) \neq 0$, $\varphi(a, E)$ is a parallel composition of sequential behaviors (generating some new parallel branches). If $\varphi(a, E) = \Delta$, then $u \parallel \varphi(a, E) = u \parallel \Delta = u$ and u remains unchanged.

Nondeterministic choice.

8. $E[a.u + a.v + w] = E[a.(u + v) + w]$. The use of left distributivity means that environment considers behavior expressions up to trace equivalence. It also means that a system uses delayed (angelic) choice.

9. $E[u + \Delta] = E[u] + E[\Delta]$. The states $E[0]$ and $E[\Delta]$ are called *terminal states of environment*. Formally the states of the form $E[0]$ are equivalent to 0, and states of a form $E[\Delta]$ are equivalent to Δ (if the $E[\Delta] = E[\Delta] + \Delta$ is added). But from the point of view of verification it is useful to distinguish syntactically different terminal states.

Parallel behaviors.

10. $E[u] = E[v] \Rightarrow E[u \parallel w] = E[v \parallel w]$. Therefore all identities for conditions and operators can be applied within the parallel composition. A component $a_1.u_1 + \dots + a_n.u_n$ of parallel composition is called *degenerated relative to the state E*, if for all operators $a_i.pt(E, a_i) = 0$ and for all conditions α_i it is true that $(E \wedge \alpha_i) = 0$. Each component that is degenerated relatively to the state E is equivalent to 0 relatively to this state.

11. $E[u] + F[v] = F[v]$, if parallel composition u contains degenerated component relative to E . So all states of environment with degenerated components are equivalent to 0.

12. $E[u + \Delta \parallel v] = E[u \parallel v] + E[v]$.

13. $E[a_1.u_1 + a_2.u_2 + \dots] = E[a_1.u_1 \parallel v] + E[a_2.u_2 \parallel v] + \dots$, if all actions a_i are different, if a_i is a condition then u_i is terminal constant, and v does not contain components degenerated relatively to the state E . The state of environment $E[u]$ is called *dead lock state*, if there are no transitions from this state, but u is not successful termination. If there is at least one degenerated component in parallel composition, then corresponding state is a dead lock state and all dead lock states are equivalent to 0, but it is useful to distinguish them as well as terminal constants. The rules (9), (12), and (13) are called *unfolding of nondeterministic choice*.

14. $E[a_1.u_1 \parallel \dots \parallel a_n.u_n] = \sum_{i=1}^n a_i.(\dots \parallel a_{i-1}.u_{i-1} \parallel u_i \parallel a_{i+1}.u_{i+1} \parallel \dots)$, if all components of parallel composition are non-degenerated. This relation is called *unfolding of a parallel composition*. This is a complete unfolding and the main result

of this paper shows that it is not needed to make the complete unfolding at each step of verification. Let be $u = a_1.u_1 \parallel \dots \parallel a_n.u_n$,

$$unfold(u, i) = a_i.(\dots \parallel a_{i-1}.u_{i-1} \parallel u_i \parallel a_{i+1}.u_{i+1} \parallel \dots)$$

The identity (14) can be rewritten as

$$14a. E[a_1.u_1 \parallel \dots \parallel a_n.u_n] = \sum_{i=1}^n unfold(u, i).$$

Environment does not distinguish trace equivalent behaviors and consequently bisimilar states of environment are trace equivalent. The identity (14) defines the main transition rule for the system:

$$E[u] \xrightarrow{a_i} E'[u'],$$

if u is a parallel composition with non-degenerated components and $E'[u']$ is defined by the identity (7).

4 Behaviors Over Basis B

The set of symbols is given for the set B of behavior basis. These symbols are called *basic sequential behaviors*. The expression of the algebra of sequential behaviors constructed from these symbols and termination constants are called *sequential behavior over basis B*. Suppose that for each symbol $v \in B$ an equation of the form $v = F_v(v_1, v_2, \dots)$ is given with sequential behavior over basis B as a right hand side. This equation is called the *definition of a basic behavior v*. The application of this definition (the substitution of the right hand side for the left hand one) is called the *unfolding of this behavior*. System of basic behaviors is called non-degenerated if each path in the tree representation of the expression $v = F_v(v_1, v_2, \dots)$ contains at least one operator.

Normal form of sequential behavior is an expression of the form $a_1.u_1 + a_2.u_2 + \dots + a_n.u_n + \varepsilon$ where u_1, u_2, \dots are sequential behaviors. If a_i is a condition, then u_i is a termination constant, $n \geq 0$, and all actions are different (not equivalent with respect to the environment E), because of delayed (angelic) choice (see sec. 2).

Each sequential behavior u over non-degenerated basis in a state $E[u]$ can be reduced to a normal form v equivalent to u with respect to E .

Parallel behavior over B is a parallel composition of sequential behaviors over B .

Normal form of parallel behavior is a nondeterministic sum of behaviors of the form $a_1.u_1 + a_2.u_2 + \dots$, where u_1, u_2, \dots are sequential behaviors over B , a_1, a_2, \dots - operators or conditions at what if a_i is a condition, then u_i is a termination constant.

Normal form of environment state is a term of a form $\sum_{i \in I} a_i.E_i[u_i] + \sum_{j \in J} \Delta$ or 0. *Each environment state with non-degenerated system of basic behaviors is trace equivalent to some normal form.*

5 Verification

A property ξ of environment state is called to be *correct* if it does not distinguish equivalent states. A property ξ of environment state is monotonic if $E \rightarrow E' \Rightarrow \xi(E[u]) \rightarrow \xi(E'[u])$.

5.1 Verification problem

For a given set Ξ of correct and monotonic checked properties, defined on the set of environment states, the set of initial states defines which properties are reachable(not reachable) from the initial states for a finite number of steps or a number of steps bounded by some constant. It is supposed that the set of checked properties contains the property of a state to be dead lock and to be a state of successful termination.

The simplest verification algorithm is exhaustive unfolding of initial states up to saturation or depletion of a given number of steps. It uses the next formulae of unfolding: $\sum_{i=1}^n E[\text{unfold}(u, i)]$. The checked properties are checked in the process of unfolding and the states satisfying checked properties are collected. More economic algorithm can be constructed using the following *partial unfolding algorithm*.

6 Partial Unfolding

Two operators a and a' are called *permutable with respect to the state E* , if $E[a * a'] = E[a' * a](a \xleftarrow{E} a')$. Let the state of environment $E[u] = E[a_1.u_1 \parallel \dots \parallel a_n.u_n]$ be given. Select a component $s = a_i.u_i$ and construct for this component the set $\text{nonp}(E, s)$ of those components $a_j.u_j, i \neq j$ such that a_i and a_j are not permutable with respect to the current state E . Obtain

$$\text{punfold}(E, u, i) = A(i) + B(E, i) + C(E, i)$$

$$A(i) = a_i.(\dots \parallel a_{i-1}.u_{i-1} \parallel u_i \parallel a_{i+1}.u_{i+1} \parallel \dots)$$

$$B(E, i) = \sum_{i \neq j \wedge (a_i, a_j) \in \text{nonp}(E, s)} a_j.(\dots \parallel a_{j-1}.u_{j-1} \parallel u_j \parallel a_{j+1}.u_{j+1} \parallel \dots)$$

$$C(E, i) =$$

$$= \sum_{k \neq i \wedge (a_k, a_i) \notin \text{nonp}(E, s) \wedge a_k \xleftarrow{E} a_w} a_k.(\dots \parallel a_{k-1}.u_{k-1} \parallel a_k \dots a_w.u_k \parallel a_{k+1}.u_{k+1} \parallel \dots)$$

An expression $E[\text{punfold}(E, u, i)]$ is called a *partial unfolding of parallel composition* by the component i (unlike the complete unfolding).

In general case, the algorithm uses dynamic permutability of operators, but it isn't optimal because using four times of pt function for each pair of the operators is required. This problem could be improved by using the notion of statically permutable operators.

6.1 Static Permutability Property

Theorem 1. If two operators $p = \alpha \rightarrow a, q = \beta \rightarrow b$ are permutable with respect to the states $E_1 = \alpha \wedge \beta, E_2 = \neg\alpha \wedge \beta, E_3 = \alpha \wedge \neg\beta$ then they are permutable for all states.

Assume the contrary that $\exists e(e[p^*q] \neq e[q^*p])$ and
 $E_1[p^*q] = E_1[q^*p], E_2[p^*q] = E_2[q^*p], E_3[p^*q] = E_3[q^*p]$. Consider
 $E_1[p^*q] = E_1[q^*p]$:

$$\begin{aligned} (\alpha \wedge \beta[p^*q] \Rightarrow pt(\alpha \wedge \beta \wedge \alpha, a)[p] = pt(\beta \wedge pt(\alpha \wedge \beta, a), b)) \wedge \\ \wedge (\alpha \wedge \beta[q^*p] \Rightarrow pt(\alpha \wedge \beta \wedge \beta, b)[q] = pt(\alpha \wedge pt(\alpha \wedge \beta, b), a)) \wedge \\ \wedge (\alpha \wedge \beta[p^*q] = \alpha \wedge \beta[q^*p]) \Rightarrow \\ \Rightarrow pt(\beta \wedge pt(\alpha \wedge \beta, a), b) = pt(\alpha \wedge pt(\alpha \wedge \beta, b), a) \Rightarrow \\ \Rightarrow pt(\beta \wedge pt(\alpha \wedge \beta \wedge (e \vee \neg e), a), b) = pt(\alpha \wedge pt(\alpha \wedge \beta \wedge (e \vee \neg e), b), a) \Rightarrow \\ \Rightarrow pt(\beta \wedge pt(\alpha \wedge \beta \wedge e, a), b) \vee pt(\beta \wedge pt(\alpha \wedge \beta \wedge \neg e, a), b) = \\ = pt(\alpha \wedge pt(\alpha \wedge \beta \wedge e, b), a) \vee pt(\alpha \wedge pt(\alpha \wedge \beta \wedge \neg e, b), a) \end{aligned}$$

Consider $E_2[p^*q] = E_2[q^*p]$:

$$\begin{aligned} (\neg\alpha \wedge \beta[p^*q] \Rightarrow pt(\alpha \wedge \neg\alpha \wedge \beta, a)[q] \Rightarrow 0) \wedge \\ \wedge (\neg\alpha \wedge \beta[q^*p] \Rightarrow pt(\beta \wedge \neg\alpha \wedge \beta, b)[p] \Rightarrow pt(\alpha \wedge pt(\neg\alpha \wedge \beta, b), a)) \wedge \\ \wedge (\neg\alpha \wedge \beta[p^*q] = \neg\alpha \wedge \beta[q^*p]) \Rightarrow \\ \Rightarrow (pt(\alpha \wedge pt(\neg\alpha \wedge \beta, b), a) = 0) \end{aligned}$$

Consider $E_3[p^*q] = E_3[q^*p]$:

$$\begin{aligned} (\alpha \wedge \neg\beta[q^*p] \Rightarrow pt(\beta \wedge \alpha \wedge \neg\beta, b)[q] \Rightarrow 0) \wedge \\ \wedge (\alpha \wedge \neg\beta[p^*q] \Rightarrow pt(\alpha \wedge \alpha \wedge \neg\beta, b)[q] \Rightarrow pt(\beta \wedge pt(\alpha \wedge \neg\beta, a), b)) \wedge \\ \wedge (\alpha \wedge \neg\beta[p^*q] = \alpha \wedge \neg\beta[q^*p]) \Rightarrow \\ \Rightarrow (pt(\beta \wedge pt(\alpha \wedge \neg\beta, a), b) = 0) \end{aligned}$$

Let's try to prove this theorem by contradiction. Let's consider insertion of two operators p, q :

$$\begin{aligned} e[p^*q] = e[(\alpha \rightarrow a)^*(\beta \rightarrow b)] \Rightarrow pt(e \wedge \alpha, a)[\beta \rightarrow b] \Rightarrow pt(\beta \wedge pt(e \wedge \alpha, a), b) \\ e[q^*p] = e[(\beta \rightarrow b)^*(\alpha \rightarrow a)] \Rightarrow pt(e \wedge \beta, b)[\alpha \rightarrow a] \Rightarrow pt(\alpha \wedge pt(e \wedge \beta, b), a) \end{aligned}$$

So, $\exists e(pt(\beta \wedge pt(e \wedge \alpha, a), b) \neq pt(\alpha \wedge pt(e \wedge \beta, b), a))$.

$$\begin{aligned} \exists e(pt(\beta \wedge pt(e \wedge \alpha \wedge (\beta \vee \neg\beta), a), b) \neq pt(\alpha \wedge pt(e \wedge \beta \wedge (\alpha \vee \neg\alpha), b), a)) \Rightarrow \\ \Rightarrow \exists e(pt(\beta \wedge pt(e \wedge \alpha \wedge \beta \vee e \wedge \alpha \wedge \neg\beta, a), b) \neq \\ \neq pt(\alpha \wedge pt(e \wedge \beta \wedge \alpha \vee e \wedge \beta \wedge \neg\alpha, b), a)) \Rightarrow \\ \Rightarrow \exists e(pt(\beta \wedge pt(e \wedge \alpha \wedge \beta, a), b) \vee pt(\beta \wedge pt(e \wedge \alpha \wedge \neg\beta, a), b) \neq \\ \neq pt(\alpha \wedge pt(e \wedge \alpha \wedge \beta, b), a) \vee pt(\alpha \wedge pt(e \wedge \neg\alpha \wedge \beta, b), a)) \Rightarrow \end{aligned}$$

So, using monotonic property of pt function obtain

$$\begin{aligned} e \wedge \neg\alpha \wedge \beta \rightarrow \neg\alpha \wedge \beta \Rightarrow pt(e \wedge \neg\alpha \wedge \beta, b) \rightarrow pt(\neg\alpha \wedge \beta, b) \Rightarrow \\ \Rightarrow pt(\alpha \wedge pt(e \wedge \neg\alpha \wedge \beta, b), a) \rightarrow pt(\alpha \wedge pt(\neg\alpha \wedge \beta, b), a) \Rightarrow \\ \Rightarrow pt(\alpha \wedge pt(e \wedge \neg\alpha \wedge \beta, b), a) \rightarrow 0 \\ e \wedge \alpha \wedge \neg\beta \rightarrow \alpha \wedge \neg\beta \Rightarrow pt(e \wedge \alpha \wedge \neg\beta, a) \rightarrow pt(\alpha \wedge \neg\beta, a) \Rightarrow \\ \Rightarrow pt(\beta \wedge pt(e \wedge \alpha \wedge \neg\beta, a), b) \rightarrow pt(\beta \wedge pt(\alpha \wedge \neg\beta, a), b) \Rightarrow \\ \Rightarrow pt(\beta \wedge pt(e \wedge \alpha \wedge \neg\beta, a), b) \rightarrow 0 \end{aligned}$$

It means that

$$\begin{aligned} & \exists e(pt(\beta \wedge pt(e \wedge \alpha \wedge \beta, a), b) \neq pt(\alpha \wedge pt(e \wedge \alpha \wedge \beta, b), a)) \wedge \\ & \wedge (E_1[p * q] = E_1[q * p]) \Rightarrow \\ & \Rightarrow \exists e((pt(\alpha \wedge pt(e \wedge \alpha \wedge \beta, b), a) = pt(\beta \wedge pt(\neg e \wedge \alpha \wedge \beta, a), b)) \wedge \\ & \wedge (pt(\beta \wedge pt(e \wedge \alpha \wedge \beta, a), b) = pt(\alpha \wedge pt(\neg e \wedge \alpha \wedge \beta, b), a))) \end{aligned}$$

Let consider the cases when

$$\exists e(pt(\alpha \wedge pt(e \wedge \alpha \wedge \beta, b), a) = pt(\beta \wedge pt(\neg e \wedge \alpha \wedge \beta, a), b))$$

$e \wedge \alpha \wedge \beta \wedge \neg e \wedge \alpha \wedge \beta = 0$ means that the pt function translates this formulae into one state independently from e and $\neg e$. It is possible then all attribute expressions from e are in r, s list of pt and after application of both protocols p, q no restrictions are left from e and $\neg e$, because of contradiction in other cases. It means that both operators p, q translate all sub-formulae which depend on attribute expressions from e into one sub-formulae, independently from sequence of application. So, by obtaining a contradiction, due to that case $pt(\alpha \wedge pt(e \wedge \alpha \wedge \beta, b), a) = pt(\beta \wedge pt(e \wedge \alpha, a), b)$, theorem is proved.

Two operators $p = \alpha \rightarrow a, q = \beta \rightarrow b$ are called *statically permutable* if they satisfy the next conditions:

1. $1. pt(\alpha \wedge pt(\alpha \wedge \beta, b), a) = pt(\beta \wedge pt(\alpha \wedge \beta, a), b)$
2. $2. pt(\alpha \wedge pt(\neg \alpha \wedge \beta, b), a) = 0$
3. $3. pt(\beta \wedge pt(\alpha \wedge \neg \beta, a), b) = 0$

From practical point of view, to make 8 calls of pt function to check the static permutability property for two operators is a slow process. So, let's define the weak property for static permutability of two operators.

Let $r(p), s(p), z(p)$ be the lists of predicate transformer for operator $p = \alpha \rightarrow a$, where $r(p)$ - the list of the attribute expressions from left part of assignment of a , $s(p)$ - the list of the attribute expressions from the formulae part of a , $z(p)$ - the list of attribute expressions from α which are not in $r(p) \cup s(p)$ [7].

Two operators $p = \alpha \rightarrow a, q = \beta \rightarrow b$ are called *weak static permutable* if

$$\begin{aligned} & ((r(p) \cup s(p)) \cap (r(q) \cup s(q) \cup z(q)) = \emptyset) \wedge \\ & \wedge ((r(q) \cup s(q)) \cap (r(p) \cup s(p) \cup z(p)) = \emptyset) \end{aligned}$$

Theorem 2. If two operators $p = \alpha \rightarrow a, q = \beta \rightarrow b$ satisfy weak condition of static permutability then they are statically permutable

If p, q satisfy weak condition of static permutability then both operators work with different memory. It means that all conditions for static permutability are satisfied and the theorem proved.

7 Algorithm of Reducing Interleaving

Using of the Breadth-first search (BFS) algorithm is better then the Depth-first search (DFS) algorithm for checking of reachability property. But, in general case, our algorithm could be used with any traversal strategy. So, let use BFS algorithm and the component which was chosen is in normal form $E[a_1.u_1 \parallel a_2.u_2 \parallel \dots], s_i = a_i.u_i$. Then

component $nonp(s_j) = \min_i nonp(s_i)$ is chosen for partial unfolding $punfold(a_1.u_1 \parallel a_2.u_2 \parallel \dots, j)$, $j \in \{1, 2, \dots\}$. Of course the algorithms for visited and dead lock detection should be defined for partial unfolding.

The state in the set of search tree is considered as visited if it is in the set of already visited states and all its possible successors are in this set:

$$E[a_1.(u_1 \parallel a_2.u_2 \parallel \dots \parallel a_i.u_i \parallel \dots \parallel a_n.u_n)] - visited$$

...

$$E[a_i.(a_1.u_1 \parallel \dots \parallel a_{i-1}.u_{i-1} \parallel u_i \parallel a_{i+1}.u_{i+1} \parallel \dots \parallel a_n.u_n)] - visited$$

...

$$E[a_n.(a_1.u_1 \parallel \dots \parallel a_{n-1}.u_{n-1} \parallel u_n)] - visited$$

The state in the set of search tree is considered as dead lock if some of inserted actions gives 0:

$$E[a_1.(u_1 \parallel a_2.u_2 \parallel \dots \parallel a_i.u_i \parallel \dots \parallel a_n.u_n)] \Rightarrow 0$$

...

$$E[a_i.(a_1.u_1 \parallel \dots \parallel a_{i-1}.u_{i-1} \parallel u_i \parallel a_{i+1}.u_{i+1} \parallel \dots \parallel a_n.u_n)] \Rightarrow 0$$

...

$$E[a_n.(a_1.u_1 \parallel \dots \parallel a_{n-1}.u_{n-1} \parallel u_n)] \Rightarrow 0$$

In general case the partial unfolding loses states, because $punfold(u, i) \neq unfold(u, i)$.

Theorem 3. The function $punfold(u, i)$ doesn't break reachability property if algorithm checks reachability property for lost states after partial unfolding.

First of all the partial unfolding doesn't break the reachability of terminal states: 0, Δ , \perp , because of definition of dead lock state, $u \parallel \Delta = u$, $u \parallel \perp = \perp$. Algorithm for visited doesn't break reachability property because of definition which helps system to stop the consideration of infinite number of states.

Finally, let's check the reachability property for goal and safety detection algorithm. The partial unfolder $punfold(u, i)$ loses the states for components of parallel composition $a_j.u_j$ which are considered as permutable for current action $(a_i, a_j) \notin nonp(s_i)$ and $\forall (a'_j \in u_j)((a_i, a'_j) \notin nonp(s_i))$. In that case $punfold$ loses states $a_j.E'[a_1.u_1 \parallel \dots \parallel u_j \parallel \dots \parallel a_n.u_n]$. It means that the algorithm should check the reachability property here. From other point of view, it's known that $E[a_i * a_j] = E[a_j * a_i]$, because $(a_i, a_j) \notin nonp(s_i)$. It means that the reachability property after insertion of a_j operator will be saved and after sequential insertion of a_i, a_j . So, the theorem is proved.

If the algorithm of partial unfolding is used for checking reachability property of goal and safety states then algorithm should check those states after each work of function $punfold$.

8 Example of Application

Let initial state be $E[R20||U312]$. The behavior $R20$ and $U312$ is defined by the following graphs fig. 1, fig. 2 respectively.

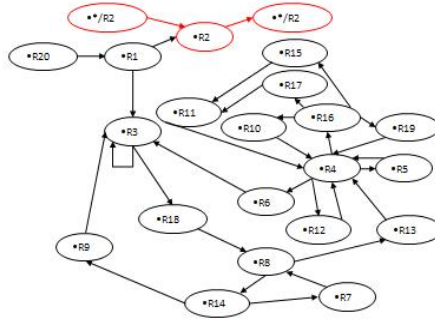


Fig. 1. Behavior for $R20$.

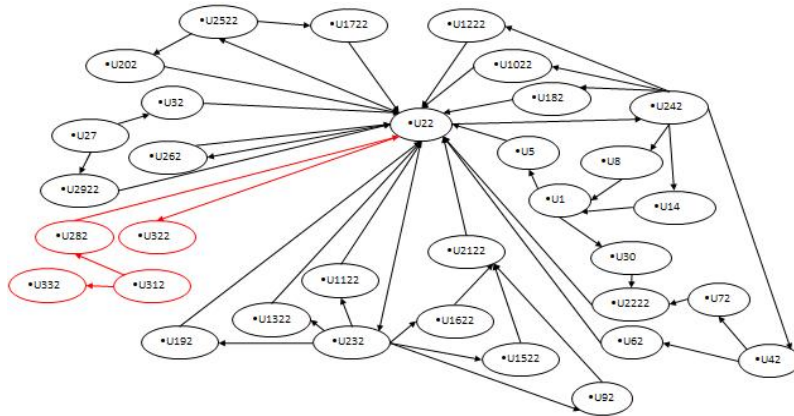


Fig. 2. Behavior for $U312$.

The $*/R2$ means all operators except $R2$, the red color in fig. 1 means the special situation when the $R2$ could be inserted after any of operators except $R2$ and that after insertion of $R2$ any protocol could be inserted except $R2$. The red color on fig. 2 used to mark sub-path for successful termination state (terminal states).

The following results are obtained after analyzing the two lists of operators from behavior $R20$ and $U312$:

1. Behavior $R20$ has 5 from 20 operators which are statically permutable for all operators from $U312$.
2. Behavior $U312$ has 31 from 33 operators which are statically permutable for all operators from $R20$.
3. The time for creating the list of statically permutable operators is aprox. 1 min.
4. The whole state space was covered after aprox. 25 min.
5. Total number of covered states is 1102.

Out of memory error was obtained without partial unfolding algorithm after aprox. 1 hour of work and aprox. 50000 of states were covered. So, it is good results for this example, because if one of operators is statically permutable for all of the operators from other parallel processes then $punfold(u,i) = A(i)$, because $B(i) = C(i) = 0$ here.

9 Conclusions

The verification algorithm based on partial unfolding has been implemented in the system of insertion modeling IMS and has shown considerable decreasing of the verification time on this example. Generally speaking, the C++ version (the language of implementation of IMS) of this algorithm will be faster not less than 10 times. It is true because of well known notion of IMS: C++ algorithm is faster not less in 10 times than corresponded prototype (which was written on APLAN language – the language of IMS) of this algorithm in IMS. We hope this algorithm will be fast for the similar examples as well.

In the near future this algorithm will be applied for the verification of set of industrial examples, for verification of parallel programs [8], and for VFS (verification of formal specification) - our tool for symbolic modeling with infinite number of states.

References

1. Escobar, S., Meseguer, J.: Symbolic Model Checking of Infinite-State Systems Using Narrowing. Proceedings of the 18th International Conference on Term Rewriting and Applications, LNCS 4533, pp. 153--168, Springer (2007).
2. McMillan, K.L.: Trace Theoretic Verification of Asynchronous Circuits Using Unfoldings. Proceedings of the 7th Workshop on Computer Aided Verification, Liege, LNCS 939, pp. 180-195, Springer (1995)
3. Esparza, J. and Heljanko, K.: Unfoldings - A Partial-Order Approach to Model Checking. EATCS Monographs in Theoretical Computer Science, ISBN: 978-3-540-77425-9, Springer-Verlag, 172 p. (2008)
4. Letichevsky, A., Gilbert, D.: A Model for Interaction of Agents and Environments. In D. Bert, C. Choppy, P. Moses, editors. Recent Trends in Algebraic Development Techniques. Lecture Notes in Computer Science 1827, Springer (1999)
5. Letichevsky, A.: Algebra of behavior transformations and its applications, in V.B.Kudryavtsev, I.G.Rosenberg (eds.) Structural theory of Automata, Semigroups, and Universal Algebra, NATO Science Series II. Mathematics, Physics and Chemistry - Vol. 207, pp. 241-272, Springer (2005)
6. Letichevsky, A., Kapitonova, J., Kotlyarov, V., Letichevsky Jr., A., Nikitchenko, N., Volkov, V., Weigert, T.: Insertion modeling in distributed system design, Problems of programming (ISSN 1727-4907), Vol. 4, pp. 13-39 (2008)
7. Letichevsky, A.A., Godlevsky, A.B., Letichevsky, A.A., Jr., Potienko, S.V., Peschanenko, V.S.: Properties of Predicate Transformer of VRS System. Cybernetics and System Analyses, 4:3-16 (2010) (in Russian)
8. Letichevsky, A., Letychevskiy, O., Peschanenko, V.: Insertion Modeling System, PSI 2011, Lecture Notes in Computer Science, Vol.7162, pp. 262-274. Springer (2011)

The Information System as a Tool to Manage R&D at the National Academy of Pedagogical Sciences of Ukraine

Natalya Zadorozhna¹, Basyl Petrushko¹, and Sergey Tukalo¹

¹Institute of information technology and learning tools,
9, M. Berlynckogo St., 04060, Kyiv, Ukraine
nzalert@list.ru, vasso@gmail.com, kolobox@bigmir.net

Abstract. This paper is devoted to the results of R&D entitled "Scientific and methodical support for an information system based on the Internet to manage R&D at the National Academy of Pedagogical Sciences of Ukraine," 0109U002139. The objective is development of technique and technology to build an information system to manage R&D. An information system towards this end was created and was launched in 2011. It is the first stage of deployment of electronic document management at the academy. This system provides documentary support to manage R&D at research institutions. It is designed as a corporate Internet portal (<http://planning.edu-ua.net>) on Microsoft Office SharePoint Server 2007 platform with added applications and document templates developed specifically to support R&D workflows in this system.

Keywords: information system, R&D Management, conceptual model, portal, MS SharePoint, electronic document management

Key Terms: Academia, Development, Management, Process, Research

1 Introduction

Implementation of information systems designed to management science and education is well timed because this contributes to the modernization of science and education in Ukraine. It corresponds to the actual for Ukraine the problem of forming a modern information society that legally defined in the Law of Ukraine "On main principles of information society development in Ukraine for 2007-2015" dated January 9, 2007 № 537. The main problem arising from this Law is to build national information systems, including information systems of management in education area.

Create an corporate information system of National Pedagogical Sciences of Ukraine (hereinafter referred to as "NAPSU") is a practical step to build similar systems.

This paper describes methodology and tools to create information system to manage R&D at the National Academy of Pedagogical Sciences of Ukraine (hereinafter referred to as "R&D Management").

2 Overall System Characteristic

The goals of this information system are to manage R&D at the National Academy of Pedagogical Sciences of Ukraine (hereinafter referred to as “R&D Management”) and the automation of documentary support for “R&D Management” in research institutions in accordance with the state regulations for research. It should provide the data and documents uploaded by "R&D Management" to officers and researchers in the Presidium and Research Institutions at the National Academy of Pedagogical Sciences of Ukraine (hereinafter referred to as “NAPSU”) based on a permissions policy with varying powers.

2.1 Goals and Objectives

The objectives of "R&D Management" are to design a system as a corporate Internet portal (<http://planning.edu-ua.net>) on the Microsoft Office SharePoint Server 2007 platform. The front-end portal is designed as a SharePoint site. The back-end portal includes the SharePoint basic tools for documents and workflows with added applications and document templates developed specifically for the "R&D Management."

"R&D Management" is the first step to create a corporate portal for electronic document management in the NAPSU. Such a system will provide conditions to migrate from a paper-based document management system to electronic document management system. This migration to an electronic document management system is well timed because the legislative and normative base is already in existence at the state level in Ukraine. Thus, "R&D Management" is a practical step to implement the state ICT policy to manage scientific and educational areas. The "R&D Management" essentially will promote the everyday use of ICT, reduce the time taken to process R&D documents, enhance the productivity of the researchers and thus be more balanced in the use of labor resources in the organization and implementation of the researches.

2.2 Subject Domain of the “R&D Management”

The main activity of the institutions of NAPSU is scientific research. Currently, research management is going through the paper documents that are sent in duplicate from research institutions to the Presidium of NAPSU. Research management at the research institutions of NAPSU is regulated by the "Statute regarding implementation of research in the National Academy of Pedagogical Sciences of Ukraine" document (released 2011). This Statute is developed under the laws of Ukraine "On scientific and scientific and technical activity," "Regarding scientific and technical examination," "On innovative activity," the state standard of Ukraine SSTU, a 3973-2000 "System of development and imputing of products in production, rules of implementation of research works," General Charter of the National Academy of Pedagogical Sciences of Ukraine and other normatively legal acts that regulate relations in a scientific field. The Statute defines the basic principles to manage

research in institutions of NAPSU and sets general requirements for R&D: applications, approval, budget, monitoring, accounting, reporting, results assessment and acceptance criteria. The action of the Statute spreads to all fundamental and applied researches that are planned and performed in institutions of NAPSU by state budget funds and other sources.

2.3 Types of Document in the “R&D management”

An R&D document is prepared in the institutes of NAPSU by R&D top managers, officials of the Academic Department and accountants:

1. R&D top managers prepare the following documents:

- R&D applications;
- Requirement Specifications for R&D;
- Description of R&D for NANU;
- Agreements of Vendors in R&D;
- Research Programs of R&D;
- Registration Cards of R&D; and
- Plan and Terms of R&D.

2. Officials of the Academic Council prepare the following documents:

- Perspective Thematic Plan of R&D;
- Annual Plan of R&D;
- Extracts from the minutes of the Academic Council Meeting:
 - to approve R&D;
 - to approve the Perspective Thematic Plan of R&D; and
 - to approve Research Programs of R&D.

3. Accountants prepare the following documents:

- Planned calculation of the estimated cost of work for each topic of R&D (includes cost estimates for all items);
- Protocol Agreement, the cost of work for each topic of R&D;
- Actual costs of the institution on a monthly and quarterly basis;
- Summary Cost estimates in all subjects of R&D.

The following supporting papers are used to complete documents that R&D top managers prepare:

- An overall picture of NAPSU;
- Resolutions of the Presidium of NAPSU;
- General description of the Departments of NAPSU;
- Protocols of the Bureau of the Departments of NAPSU;
- Log of R&D;

- Overall about Research Institutions of NAPSU;
- Staff list of the Research Institution of NAPSU.

2.4 “R&D Management” Requirements

"R&D Management" should support the comfortable environment of collective activity to manage R&D and effective automated procedures of management documents (forming, access and synchronization of changes). Thus "R&D Management" belongs to a class of systems of electronic document management (EDM) [1-2].

The defined feature of this class is to provide the document management and group work with documents. A management document requires the specific processing procedures corresponding to the specific document types and the tools to support the depository documents. Group work with documents should be determined by politics of permission and roles of users.

These fundamental requirements for electronic document management define the main design decisions of "R&D Management" [2].

3 Design Decisions “R&D Management”

The data structure of "R&D Management" is specified as the set of fields of all types of R&D documents. To automatically fill out those documents, it is necessary to specify a set of the same fields (hereinafter named common fields), rules and sequences to process every common field in every document. The field in the document is filled by users manually, its value should be chosen from a set list, or the system should calculate values using fields, specified in supporting papers (financial performance, staff list, etc.). Supporting synchronic changes in common fields in all the documents is very important for "R&D Management".

3.1 A Conceptual Data Model

A list of common fields for all groups of documents was systematized and formalized as the conceptual data model of "R&D Management," which is described in the terms of subject domain [2].

The conceptual data model "R&D Management" being built is as follows:

1. To analyze R&D Management in NAPSU;
2. To compose lists of common fields in the documents, i.e., the same fields that are used in more than one document;
3. To identify fields;
4. To determine the source document for each common field;
5. To define documents that use each common field;
6. To determine and describe a procedure to fill each common field; and
7. To define fields whose values are selected from the appropriate set list.

The conceptual data model "R&D Management" is formally described as a table with the following column: Document Name, ID (Document Identifier), Owner (Department name where the document is prepared), Common Fields (consisting of two columns: Field Name and Field ID).

A unique identifier is appropriated for every document. A field identifier consists of a document identifier and a sequence number of the field in the source document. An identifier color is selected depending on the group. The following field groups are found in the conceptual data model:

- a field-source (it is filled out in this document, and its value is used in all other documents, so it is required unconditionally to synchronize a common value field through all documents);
- a field-source (it is filled out in this document, and this value is used in some documents, and thus required to synchronize a common value field in some documents);
- a copy of the field-source;
- the field is calculated automatically; the field is used to calculate other fields; and
- a check box.

Table of conceptual model "R&D Management" describes a total 147 documents and 253 common fields.

3.2 MS SharePoint – Software Platform "R&D Management"

As noted above, "R&D Management" belongs to a class of systems of EDM.

Microsoft SharePoint Products and Technologies (hereinafter referred to as SharePoint), namely Microsoft Office SharePoint Server 2007, is adopted as the "R&D Management" platform due to the above EDM requirements.

The main factor for adopting SharePoint as a software platform for "R&D Management" is a familiar working environment (as a rule it is MS Office package and program browser), to which the user is accustomed in his/her daily activities. SharePoint integrates with Microsoft Office, which allows a largely familiar environment to be saved for the user. Moreover, the system developed on SharePoint platform does not require installing any additional client software or special user knowledge in information technology. To use this system, the user needs to have operating systems such as Windows (Microsoft Windows 98, Windows Millennium Edition, Windows XP and above), and a package of office applications (Microsoft Office 97 and above).

SharePoint also supports centralized repository of documents (SharePoint library), a mandatory component of any EDM.

In addition, SharePoint provides a lot of features to process documents. SharePoint has a Ukrainian localization. Sites based on SharePoint allow users to share their activity. They facilitate the interaction of web applications like Wikis and blogs and create a safe environment interaction. Their content management system is effective and usable. A Web Part, also called a Web Widget, is an ASP.NET server control which is added to a Web Part Zone on Web Part Pages by users at run time. The

controls enable end-users to modify the content, appearance, and behavior of web pages directly from a browser. It can be put into certain places in a web page by end-users, after development by the programmer. Therefore, one of the most widespread and everyday tasks of a developer on the SharePoint platform is the development of Web Parts. It is a flexible and accessible method to expand the functionality of the platform (regardless of whether it will be service of site news or difficult systems of accounting and document workflow, they can be easily and flexibly customized).

A Web Part can be easily encapsulated and removed. Nowadays, the trend of development for SharePoint is to create Web Parts, using custom controls and other similar tools. Those Web Parts can be added to any page in any combination, and the administrator has tools to customize the page if it is needed. Users have tools to operate with data in Lists and Document Libraries, using Controls and Web Parts.

Thus, SharePoint as software platform "R&D Management" provides a comfortable procedure for creation of a portal, use of built-in instruments for the base functions of document flow, and also tools to develop and integrate applications for R&D Management [3].

On the one hand, "R&D Management" has been developed by using standard SharePoint Services tools to create the Web Application. They are Web Parts, Lists, Document Library, Workflow and Website Template.

But on the other hand, there is an unusual implementation of SharePoint in "R&D Management"; which is a novel approach to processing documents. SharePoint is a web application platform. It supports work of its own web application ("R&D Management" is a SharePoint web application) through web server MS IIS. MS IIS gives access to data and content, stored in SQL database. In this way "R&D Management" is able to process data-only SharePoint objects at web level. Because SharePoint considers a document as an independent single object with inside data, it has no tools to control the data integrity of different documents with one and the same data in the SharePoint library. Support of data integrity, however, is an essential condition for business processes in "R&D Management." To solve this problem, special tools were developed; namely, features to synchronize data of documents in the "R&D Management" library.

Documents in "R&D Management" contain various shared tables. To process table data as express fields in terms of MS Office, special additional tools were developed in "R&D Management" because they were absent in SharePoint.

3.3 Basic Principles of the "R&D Management"

"R&D management" supports operating R&D document flow in NAPSU through the Internet portal with an access authorized user. Access to menus and services in the portal is only for registered users with permissions, which are defined by their official powers in NAPSU and administrative subordination in the NAPSU (Leadership, Respective Departments and Research Institutions).

Documents are created and stored in the "R&D management" Library. Each type of document is described as a specific Content Type Template. The specific document is created according its Content Type Template and stored in the appropriate folder of the hierarchical tree structure, which reflects administrative subordination in NAPSU.

The values of the common fields in various documents are changed synchronically with the change of the field source according to rules, which are defined by the field group. To work with documents (create, view, edit) the user opens the document in the portal using the appropriate application of the MS Office package (MS Word, MS Excel, etc.) and performs the necessary operations in documents in the usual manner. The only difference between the document opened in the portal and the document opened locally is that the documents in the portal already contain data in those common fields, which had been already filled out in the source documents. For example, the document "Application R&D" for specific R&D contains the value "Scientific and methodical support for an information system based on the Internet to manage R&D at the National Academy of Pedagogical Sciences of Ukraine" in the field "Title R&D." When each of more than 20 documents for this R&D will be open, the field "Title R&D" will contain the above value. Moreover, when it is necessary to change the value in this field (often by the advice of the Academic Council officials, expert conclusions, etc.), then change the field "Title R&D" only in the Document "Application R&D". Completing this feature is based on a function of express blocks in Word 2007 to define reusable fields within a document and add them to specially created template express blocks. Once the content is added as an express block, the user gets access to it for reuse in any Word 2007 document. Synchronization values in express fields (We named them "common fields") in various documents in "R&D Management" provides by special application developed in C#.

3.4 Sequence of Projecting

"R&D Management" consists of two parts: external, it is actually portal – front-end portal, and functional part – back-end portal.

Back-end portal. The sequence for projecting the back-end portal contains the following steps: analysis, requirement definition, design, and developing. The conceptual model is the result of analysis. The conceptual model is the base for requirement definition that determines templates and documents processing rules for data processing and workflows. Data processing is provided by using SharePoint features and additional applications to synchronize the alphanumeric fields, table synchronization, and data calculations. Additional applications implement synchronization handlers in C# and processors of the XML configuration file. A diagram of analysis and design of "R&D Management" is shown in Fig.1-4.

The main task for projecting the "R&D Management" back-end portal is the synchronization of documents; namely, synchronous modification of the common fields in all documents when the value in the field-source is changed. The UML-sequence diagram of "R&D Management" applications is shown in Fig.5.

Front-end portal. Sequence for projecting of «R&D Management» front-end portal is shown in Fig. 6.

3.5 Current Status and Prospects of “R&D Management”

R&D Management" version 1.0 was launched in 2011. The R&D documents for the 2012–2014 of Research Institutions of the Department of General Secondary Education NAPSU have been uploaded and processed in the system. The implementation received a positive evaluation. The entire "R&D Management" is to be implemented in 2012. Current version of this system is the first step to manage start to automation R&D in NAPSU. Further development of "R&D Management" consists of expansion of functionality for support of processes of R&D monitoring principles and integration with electronic library of NAPSU.

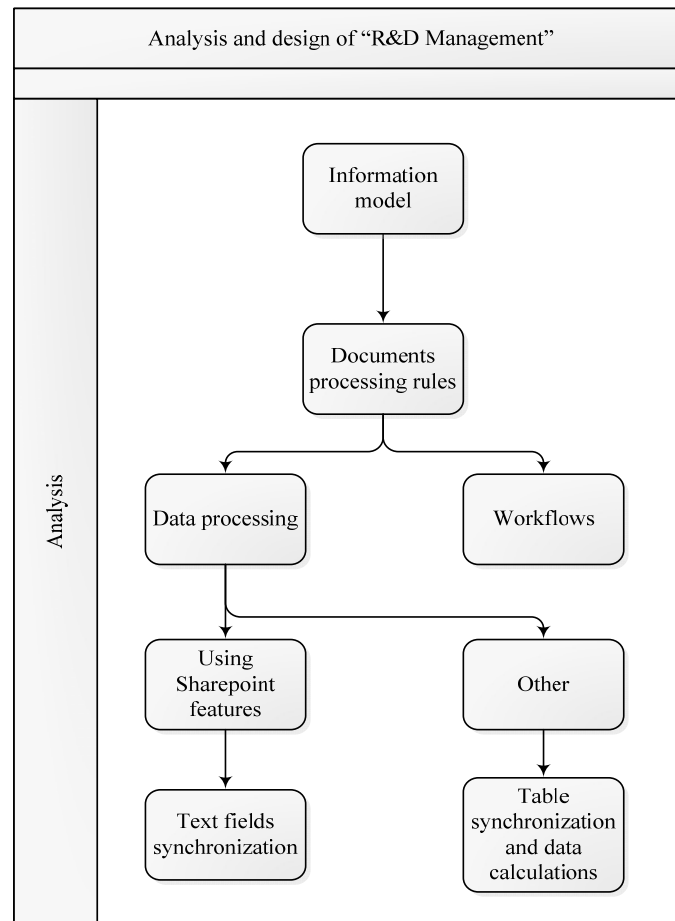


Fig. 1. Analysis and design of “R&D Management” (document processing).

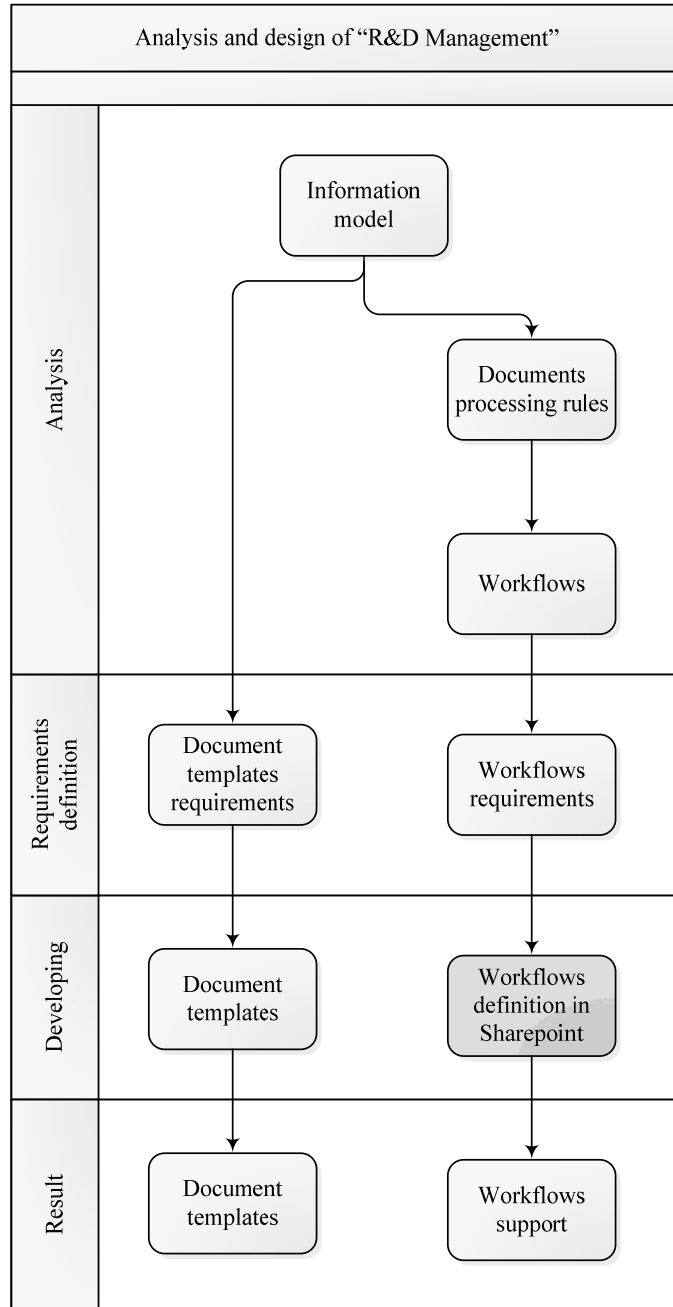


Fig. 2. Analysis and design of "R&D Management" (template & workflow).

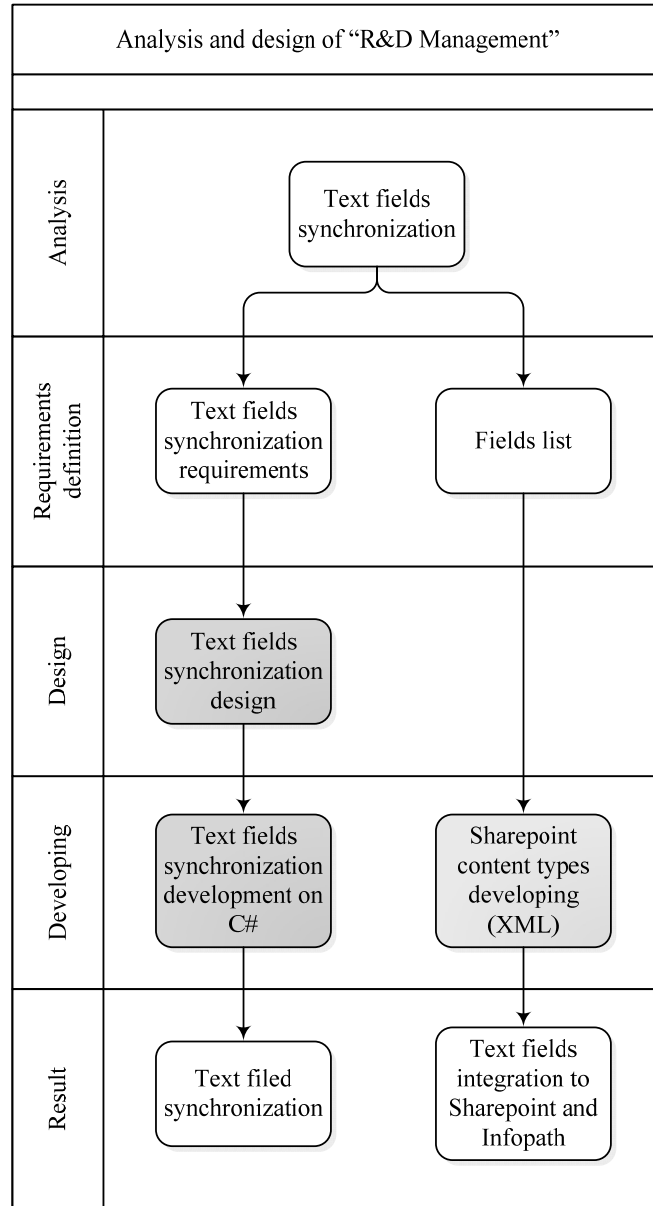


Fig. 3. Analysis and design of "R&D Management" (text field synchronization).

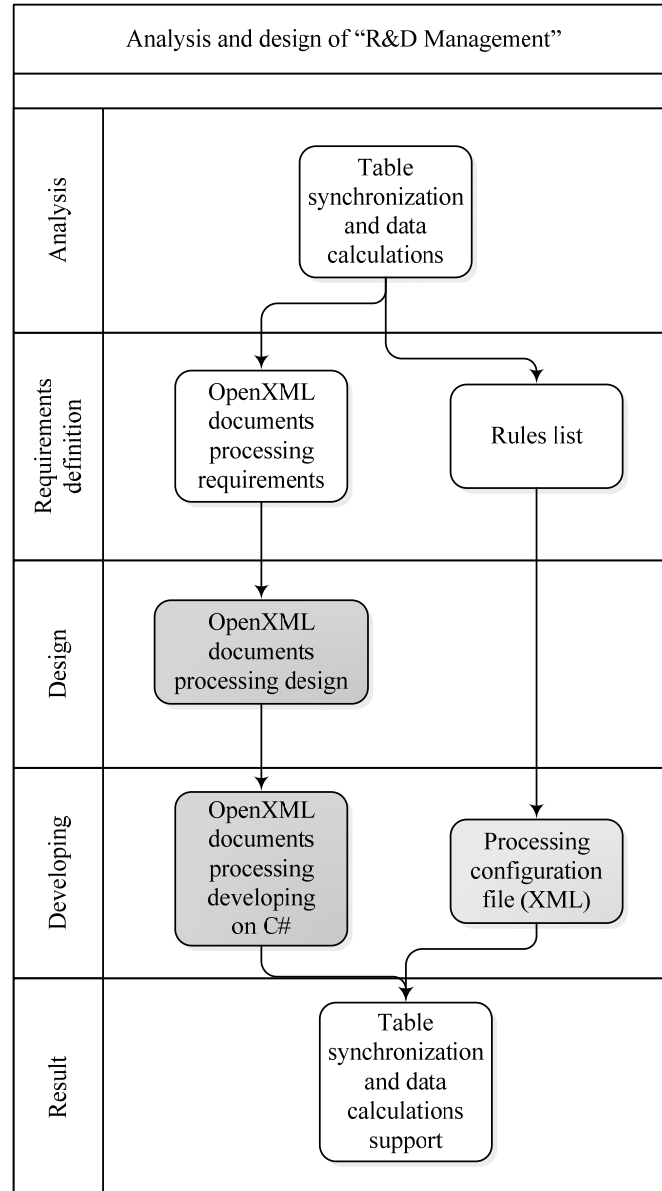


Fig. 4. Analysis and design of "R&D Management" (table and data calculation).

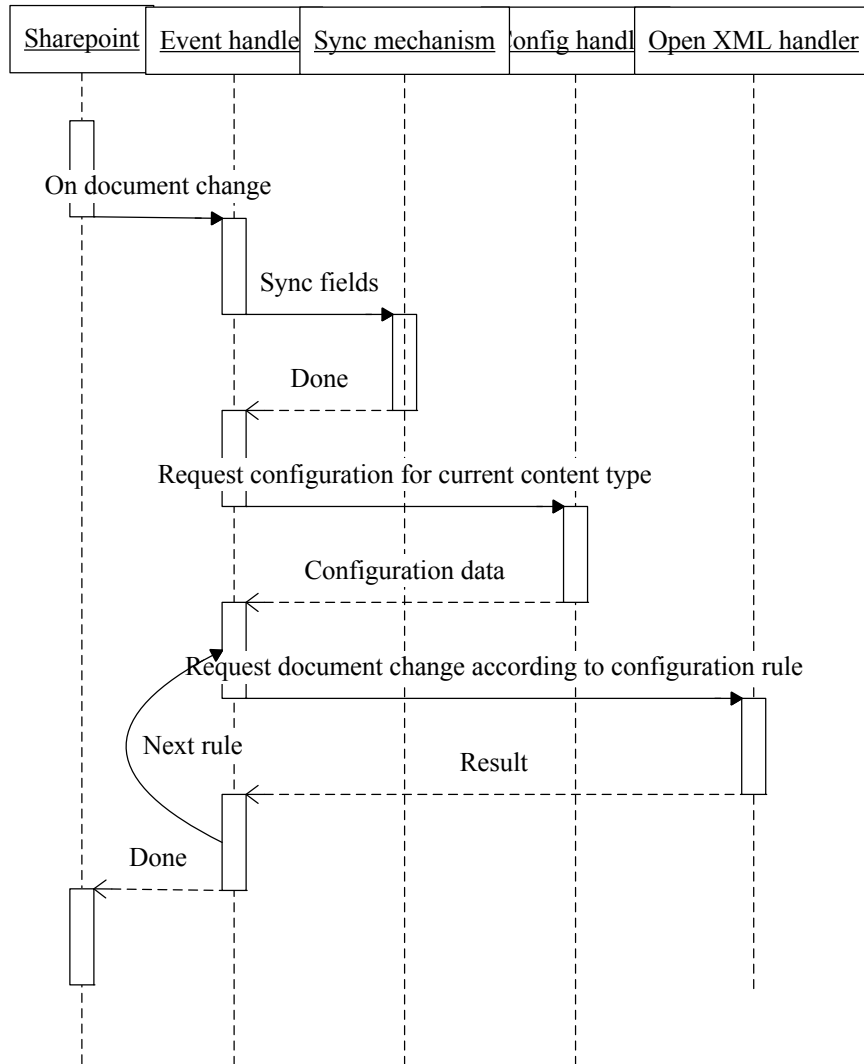


Fig. 5. UML-sequence diagram of “R&D Management” applications

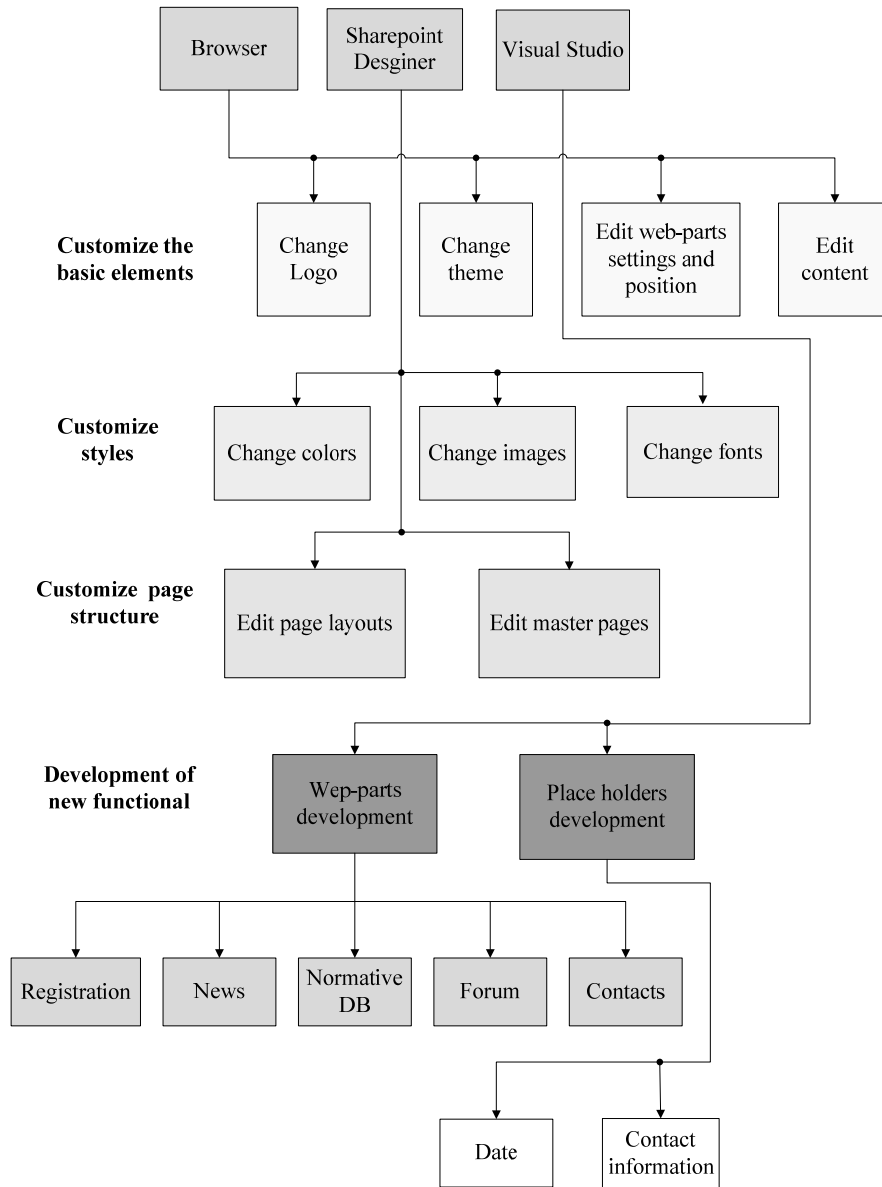


Fig 6. Sequence of projecting of “R&D Management” front-end potral

4 Conclusion

“R&D Management” is platform to deploy the electronic document management corporate portal of NAPSU for the whole spectrum of functions in document flow.

Further development of "R&D Management" consists of expansion of functionality of the corporate portal NAPS of Ukraine.

Design decisions of the “R&D Management” can be used for creation of similar systems in establishments which perform R&D with state budget funds and other sources.

Developed for “R&D Management” tools to synchronize data of documents in the SharePoint library as well as tools to process shared table data as express block (in terms of MS Office) can be used for development SharePoint web application to process shared documents.

Design decisions and actual “R&D Management” are useful as learning tool for special course “Electronic Document Management”.

References

1. Zadorozhna, N, Lavrisheva K.: Document management in the information systems (for Universities & Postgraduate Teacher Education Institutions). Pedagogichna Dumka, Kyiv (2007) (in Ukrainian)
2. Zadorozhna, N.: Concept to create the information system based on Internet to manage R&D in the National Academy of Pedagogical Sciences of Ukraine. Information Technologies and Learning Tools, Kyiv (2009) (in Ukrainian),
<http://journal.iitta.gov.ua/index.php/itlt/article/view/45/31>
3. <http://journal.iitta.gov.ua/index.php/itlt/article/view/45/31>
4. Petrushko V.: Development of Tools to Synchronize Data Documents in the SharePoint Library. Information Technologies and Learning Tools, Kyiv (2010) (in Ukrainian)
<http://journal.iitta.gov.ua/index.php/itlt/article/view/363/320>
5. Tukalo S.: Automatic processing of documents in the information system to manage R&D in the National Academy of Pedagogical Sciences of Ukraine. Information Technologies and Learning Tools, Kyiv (2010) (in Ukrainian)
6. <http://journal.iitta.gov.ua/index.php/itlt/article/view/366/323>

Maintainability Metrics of UML Design

Iryna Zaretska¹ and Maryna Besedina¹

¹V.N. Karazin Kharkiv National University, Kharkiv, Ukraine
zar@univer.kharkov.ua, reencka@rambler.ru

Abstract. The paper introduces a new object-oriented metric to evaluate maintainability of the software system at the design stage. Unlike well known object-oriented metrics applicable only to one class or to a category of several interconnected classes the proposed one evaluates the degree of extensibility for the whole static design. The metric is based on the main principles of object-oriented design and can be used by designers for evaluation and refining purposes. The results of the experiment on real projects to check this metric are reported. The Java plug-in for calculating this metric in UML Case tools is presented.

Keywords: Software design, object-oriented approach, software maintainability, object-oriented metrics.

Key Terms: SoftwareSystem, SoftwareComponent, Object, Model, Metric.

1 Introduction

The importance of evaluating the quality characteristics of the software under development at the early stages of the software lifecycle is well known in software engineering. One of such characteristics much valued at present times is its maintainability, which according to the ISO / IEC 9126 standard means “the capability of the software product to be modified. Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications.” [1]. So the main task of a designer of an object oriented software system is to produce the design easily adaptable to new or changed requirements without considerable or even any changes in the already existing coded and tested implementation. How to calculate the degree of the future system’s maintainability by the UML design?

There exist a lot of object oriented metrics to evaluate the quality of the UML design but they are limited to evaluating its complexity, or using object oriented paradigm or reusability of a class or a group of coupled classes. Neither of these metrics can tell how much effort will be needed to enhance the design in order to adapt it to new requirements. Here we restrict such enhancement to using inheritance and polymorphism so that changes in the requirements can be satisfied only by creating new classes derived from already present in the design and overriding their

methods or by creating new implementations of the already present in the design interfaces.

We introduce a metric, we call it E (from “Enhancement”), which is calculated by a class diagram and shows how easily new classes can be added to the design. In fact it uses well known in object oriented development main principle which recommends “programming to interfaces, not to implementation” [2]. It is up to the designer to decide how to use its value and whether to reconsider the design taking into account the application domain and possible future enhancements of the system.

2 Related Work

There exist a lot of object oriented metrics and sets of metrics to evaluate the quality of software design. They are divided into categories and used for different purposes. For example, Java Eclipse provides quite a big number of such metrics calculated by the project’s code. Despite the variety of these metrics the question of their values interpretation remains open. Usually it is more qualitative then quantitative interpretation like “the less the better” or “the closer to 1 the better”. It always depends on the application domain and the scope of the project how close to 0 or to 1 is good enough. It is even harder to interpret the metrics with absolute values such as DIT (Depth of Inheritance Tree) or NOC (Number Of Children). Several case studies [5, 6] report finding the most valued metrics and their weighted integral characteristic to estimate some special quality characteristics of the software system.

The closest metrics to our studies are the following ones [3, 4]. The instability metric $I = Ce / (Ca + Ce)$, where Ca - stands for a number of Afferent Couplings, Ce - stands for a number of Efferent Couplings. This metric evaluates the instability of a category of classes: $I=0$ means absolutely stable (reusable) category while $I=1$ means impossibility to reuse the category as a whole. The abstractness metric $A = nA / nAll$, where nA is a number of abstract classes in a category and $nAll$ is an overall number of classes in this category also is applicable to the category of classes. The equality $A=1$ means that the category is completely abstract and should be enhanced by inheritance to be used in a “live” software while $A=0$ shows a completely concrete category which is not good for enhancement purposes. In fact these two metrics work together forming the so called main sequence – a straight line given by the equation $A+I=1$ (Fig.1). This line defines the categories with the best balance between abstractness and instability.

Two more metrics calculated as the distance from this line $D=|(A+I-1)/\sqrt{2}|$ and the normalized distance from this line $Dn=|A+I-2|$ are also used to evaluate the categories of classes. These metrics could be used to some extent to evaluate the degree of maintainability of the category but not of the whole system. Besides some efforts are needed to automatically allocate categories and calculate these metrics.

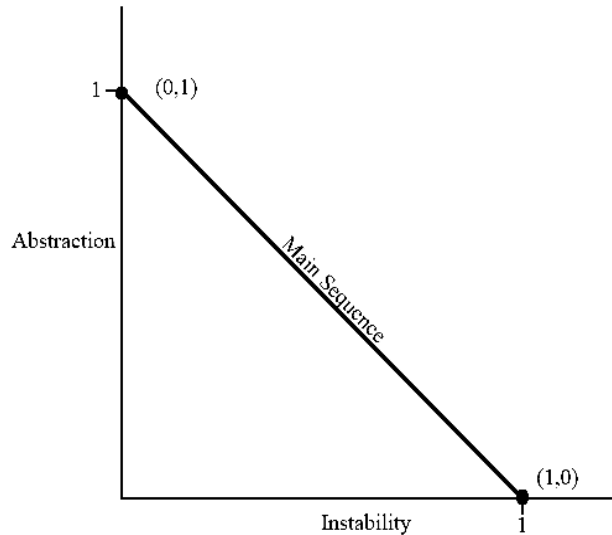


Fig. 1. Main sequence.

3 Maintainability Metrics

To create really object oriented design one has to follow several simple principles [2]:

- use interfaces to define common responsibilities of classes;
- declare variables to be instances of the interfaces, not instances of particular classes;
- use creational patterns to associate interfaces with implementations;
- keep classes focused on one task;
- if some responsibility of a class could change in future, create a separate interface to declare this responsibility and its present implementation, and use delegation technique.

Our maintainability metric is based exactly on these principles. We calculate the overall number of connections between classes on the class diagram and denote it by nC , then we distinguish those from them which connect a class to an interface or an interface to an interface, their number we denote by nI . Their ratio shows the degree to which the principles mentioned above are satisfied.

So we introduce the metrics calculated on the class diagram by the formula $E = nI / nC$, where nI is a number of the "class - interface" or "interface - interface" connections and nC is the overall number of connections. If $nI = nC = 0$ (no connections at all) we put $E = 0$.

The connection of the "class - interface" type can be as follows:

- association between a class and an interface (usually means composition);,
- implementation of an interface by a class;
- dependency between a class and an interface (usually means local visibility: a parameter of a method or the return value or the local variable of the method);
- aggregation of an interface in a class (the “whole-part” relations, containers, etc.)

As usual for such metrics its value is between 0 and 1: the closer to 1 the better. Let us see how it works in a simple well known example with validation of data. If we put the responsibility of the data validation onto a class (Fig. 2) any changes in validation rules will require changes in this class. Here $E=0$.



Fig. 2. Class Product is responsible for data validation.

But if we consider the main principles mentioned above the design will look otherwise (Fig.3).

Now $E=(2+1)/3=3/3=1$ which is the best value. New rules of validation will require new implementations of the base interface without any changes in the already existing design.

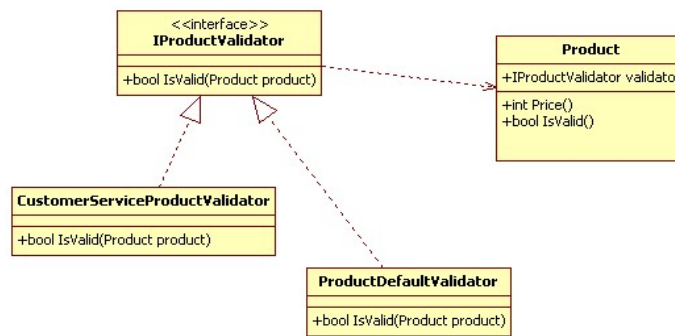


Fig. 3. Responsibility for data validation is delegated to another class.

To be certain we conducted an experiment with five middle sized projects of different scope but all developed inside University. We engaged experts to assess their quality by the following scale: 1 – needs considerable redesigning to enhance; 2 –needs small changes in existing design to enhance; 3 – no need to change existing design to enhance. Then we calculated E metric. The results are given in a Table 1.

The resulting value of E metric is useful for the designer just to ask himself if all the principles of object oriented design satisfied and if not to find reasonable grounding in the concrete application domain or/and requirements.

Table 1. The results of experiment.

Name of the project	Overall number of classes in the design model	Overall number of interfaces in the design model	Expert evaluation	Value of E metrics
Bug Tracker	20	3	1	0,143
Tester	71	7	2	0,527
Timetable	11	0	1	0
Dean's office	43	5	2	0,51
Preparatory Department	46	17	3	0,64

4 Java Plug-in for Calculating E Metrics

To make the calculation of the E metric for big projects simple we developed the Java plug-in, we called it EParser, which can be easily added to many UML CASE tools with open source. We tried it with Eclipse. The main idea is to parse the XMI file of the class diagram generated by a UML CASE tool, find there elements (in fact connections) we are interested in and make needed calculations. The class diagram of the EParser is given in Fig. 4. A simple window shows the result (Fig. 5).

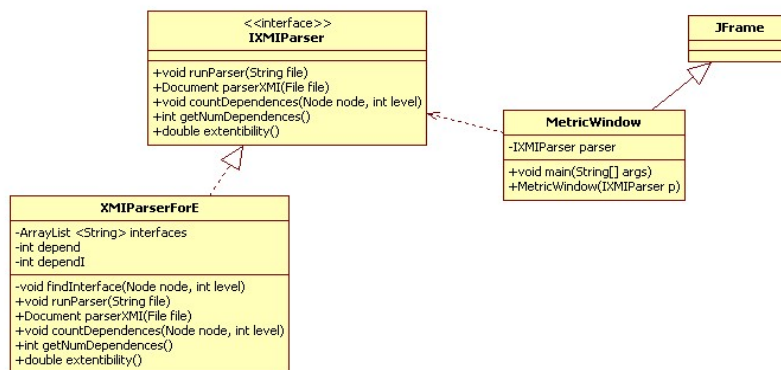


Fig. 4. Class diagram for EParser.

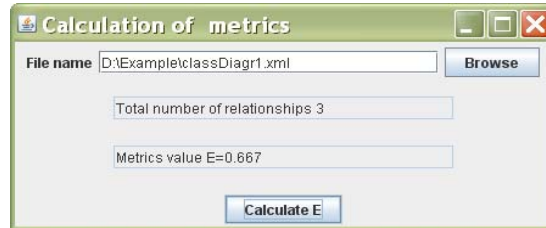


Fig. 5. Snapshot of working plug-in.

5 Conclusions

We introduced the metric that shows the degree to which the main principles of object oriented design are satisfied. As the object-oriented style provides for flexibility, reusability and maintainability of the software, this metric serves the same purpose. Certainly it is only partial indicator as it is calculated only by static decomposition of the system and does not consider its dynamic aspects. However it can be useful reminder to the designer to think about possible future changes of the system and check if the design is ready to adopt them easily and naturally.

References

1. ISO/IEC TR 9126-3:2003 Software engineering — Product quality — Part 3: Internal metrics (2003)
2. E.Gamma, R. Helm, R. Johnson, J. Vlissides: Design patterns: Elements of reusable object oriented software. Addison-Wesley (2001)
3. R.C. Martin: Designing Object-Oriented C++ Applications Using the Booch Method. Prentice-Hall (1995)
4. R. C. Martin: OO Design Quality Metrics. An Analysis of Dependencies (1994)
5. L. C. Briand, J. Wust, H. Lounis: Replicated case studies for investigating quality factors in object-oriented design. . Empirical Software Engineering. Vol. 6, Issue 1, Springer Netherlands, 11--58 (2001), <http://www.scopus.com/>
6. L. C. Briand, J. Wust, J. Daly, D. V. Porter: Exploring the relationship between design measures and software quality in object-oriented systems. Journal of Systems and Software, Volume 51 Issue 3, Elsevier Science Publishing Company, Inc., 245--273 (2000)

Combinatorial Strand Algebra in Insertion Modeling System

Dmitriy M. Klionov¹

¹Kherson State University, 40 rokiv Zhovtnya st. 27, Kherson, Ukraine
soulslayermaster@gmail.com

Abstract. This article is focused on the Insertion Modeling System developed by A.A. Letichevsky of the department 100/105 of the Glushkov Institute of Cybernetics, National Academy of Science of Ukraine, Kyiv, Ukraine. Insertion Modeling System (IMS)[1] is built on the Algebraic Programming System (APS) that also was developed by A.A. Letichevsky in 1987. and on the way of implementation of strand algebras – a process algebra for DNA computing devised by Luca Cardelli in order to compile other formal systems into the algebra, and compilation of this algebra into DNA structures. We focus on the basic strand algebra – combinatorial strand algebra, which is equivalent to the place-transition Petri nets, and on the version of the model driver of the Insertion Modeling System, based on the Petri nets.

Keywords: insertion modeling, system biology, strand algebras

Key Terms: Computation, Model, InsertionModeling, Algebra.

1 Introduction

DNA technology is reaching the point where one can envision automatically compiling high-level formalisms to DNA computational structures [11]. There are three compilation processes for concurrent languages, described by Cardelli in paper[10]. First, is the compilation of a low-level combinatorial algebra to a certain class of composable DNA structures [12]: this is intended to be a direct (but not quite trivial) mapping, which provides an algebraic notation for writing concurrent molecular programs. Second, is the compilation of a higher-level expression-based algebra to the lower-level combinatorial algebra, as a paradigm for compiling expressions of arbitrary complexity to 'assembly language' DNA combinators. Third is translating concurrent interacting automata [13] to molecular structures. There is no clear way to implement such system, because one must decompose concurrent communication patterns into a form suitable for molecular interactions (a quadratic process that is described in [13]), and then one must find some suitable 'general programmable matter' as a physical substrate. Some solution of this problem, based on the combinatorial DNA algebra, was given by Cardelli in paper[10].

Process algebras are formal languages designed to describe and analyze the concurrent activities of multiple processes. The standard technical presentation of process algebras was initially inspired by a chemical metaphor [14], and it is therefore natural, as a tutorial, to see how the chemistry of diluted well-mixed solutions can itself be presented as a process algebra. Having chemistry in this form also facilitates relating it to other process algebras.

Take a set C of chemical solutions denoted by P, Q, R . Two binary relations are defined on this set. The first relation, mixing, $P \equiv Q$ is an equivalence relation: its purpose is to describe reversible events that amount to 'chemical mixing'; that is, to bringing components close to each other (syntactically) so that they can conveniently react by the second relation. Its basic algebraic laws are the commutative monoid laws of $+$ and 0 , where $+$ is the chemical combination symbol and 0 represents the empty solution. The second relation, reaction, $P \rightarrow Q$, describes how a (sub-) solution P becomes a different solution Q . A reaction $P \rightarrow Q$ operates under a dilution assumption; namely, that adding some R to P does not make it then impossible for P to become Q (although R may enable additional reactions that overall quantitatively repress by interfering with P). The two relations of mixing and reaction are connected by a rule that says that the solution is well mixed: for any reaction to happen it is sufficient to mix the solution so that the reagents can interact. In first instance, the reaction relation does not have chemical rates. However, from the initial solution, from the rates of the base reactions, and from the relation \rightarrow describing whole-system transitions, one can generate a continuous time Markov chain representing the kinetics of the system. In terms of system evolution, it is also useful to consider the symmetric and transitive closure, \rightarrow^* , representing sequences of reactions.

As process algebra, chemistry therefore obeys the following general laws, shown lower:

$$P \equiv P; P \equiv Q \Rightarrow Q \equiv P; P \equiv Q, Q \equiv R \Rightarrow P \equiv R \quad (1)$$

Equivalence

$$P \equiv Q \Rightarrow P + R \equiv Q + R \quad (2)$$

Congruence

$$P + Q \equiv Q + P; P + (Q + R) \equiv (P + Q) + R; P + 0 \equiv P \quad (3)$$

Diffusion

$$P \rightarrow Q \Rightarrow P + R \rightarrow Q + R \quad (4)$$

Dilution

$$P \equiv P', P' \rightarrow Q', Q' \equiv Q \Rightarrow P \rightarrow Q \quad (5)$$

well mixing

Algebra is about equations, but in process algebra equations are usually a derived concept. Instead of axiomatizing a set of equations, we can use the reaction relation to study the equations that hold in a given algebra, meaning that $P = Q$ holds if P and Q produce the same reactions [15]. The complexity of these derived equational theories varies with the algebra. A simple instance here is the equation $P + 0 = P$, whose validity requires verifying that in definition of \rightarrow there is no reaction for 0, nor for 0 combined with something else.

This way, chemistry can be presented as process algebra. But the algebra of chemical '+' is one among many: there are other process algebras that can suit biochemistry more directly [16,17] or, that can suit DNA computing. In the same way the strand algebra represent DNA strands, DNA gates and operations among them allowing the higher-level formalisms to be compiled to the DNA structures. In this paper we will show the way to represent the simplest strand algebra – combinatorial strand algebra as insertion model for the Insertion Modeling System[1], using the fact that combinatorial strand algebra is equivalent to place transition Petri nets. There is a representation of Petri nets, given by A.A. Letichevsky in form of insertion machines. We use this representation in order to build a specified analytical model driver for the combinatorial strand algebra.

2 Insertion Modeling System

2.1 The Architecture of Insertion Modeling System

Insertion Modeling System(IMS) [1] developed by A.A. Letichevsky of the Glushkov Institute of Cybernetics, National Academy of Science of Ukraine, Kyiv, Ukraine. Insertion modeling is the technology of system design founded on the theory of interaction of agents and environments. It is based on process algebra and is intended for the unification of different models of interaction and computation (such as CCS, CSP, π - calculus, mobile ambients etc.).

Insertion model of a system represent this system as a composition of environment and agents inserted into it. The insertion function is usually denoted as $E[u]$ were E is the state of environment and u is the state of an agent. $E[u]$ is a new environment state after insertion an agent u. All agents and environments are labeled or attributed transition systems (labeled systems with states labeled by attribute labels [9]). The states of transition systems are considered up to bisimilarity. The main invariant of bisimilarity is the behavior $beh[E]$ of transition system in the state E (an oriented tree with edges labeled by actions and nodes labeled by attribute labels). Behaviors themselves can be considered as states of transition systems.

The general architecture of insertion machine is represented on the fig. 1.

The main component of insertion machine is model driver, the component which controls the machine movement along the behavior tree of a model. The state of a model is represented as a text in the input language of insertion machine and is considered as an algebraic expression. The input language include the recursive definitions of agent behaviors, the notation for insertion function, and possibly some

the industry standards such as UML activity diagrams, BPMN and EPCs, Petri nets offer graphical notation for stepwise processes that include choice, iteration, and concurrent execution. Unlike these standards, Petri nets have an exact mathematical definition of their execution semantics, with a well-developed theory for process analysis.

A Petri net consists of places, transitions, and arcs. Arcs run from a place to transition or vice versa, newer between places or between transitions. The places from which an arc runs to a transition are called the input places of the transition; the places to which arcs run from a transition are called the output places of the transition. Places in a Petri net may contain a discrete number of marks called tokens. Any distribution of tokens over the places will represent a configuration of the net called a marking. In abstract sense relating to Petri nets diagram, a transition of a Petri net may fire whenever there are sufficient tokens at the start of all input arcs; when it fires, it consumes this tokens, and places them at the end of all output arcs. A firing is atomic, i.e., a single non-interruptible step.

Execution of Petri nets is nondeterministic: when multiple transitions are enabled at the same time, any of them may fire, so multiple tokens may be represented anywhere in the net (even in the same place). Petri nets are well suited for modeling the concurrent behavior of distributed systems.

Petri nets are formally defined as a state-transition systems that extend a class of nets called elementary nets. Formal definition is represented lower.

1. P is a set of states, called places.
2. T is a set of transitions
3. F where $F \subset (P \times T) \cup (T \times P)$ is a set of relations called arcs
4. $N = (P, T, F)$ is a net
5. C is such that $C \subseteq P$ is a configuration
6. M so that $M : P \rightarrow Z$ is a place multiset, where Z is a countable set.
7. W so that $W : F \rightarrow Z$ is an arc multiset
8. $PN = (N, M, W)$ is a Petri net

Fig 2. Formal definition of Petri net.

Petri net is bipartite graph, where P is one partition and T is the other. Moreover, for every t in T there exist p and q in P so that (p, t) and (t, q) are in F , and for every p and q in P , if (p, t) and (t, q) are in F then $p \neq q$.

The set are the new elements. The set of places define the local states of a net, however, the global state of a net can be defined by place subsets.

In order to represent Petri nets as a composition of agents and environments, we represent transitions T as actions, tokens as agents, and places as states. The behavior of tokens located in the enabled place is written as:

$$u(s) = \sum_{W(s,t) > 0} t \cdot \Delta \quad (6)$$

The states of Petri environment are equal to the marks (configurations) of the net.

$$M : S \rightarrow Nat \Leftrightarrow E(M) = \parallel \{u(s)^{|M(s)|} \mid s \in S\} \quad (7)$$

The insertion function is defined as:

$$E(M)[u] = E(M) \parallel u \quad (8)$$

The environment transitions are defined as:

$$\frac{u \xrightarrow{t} u'}{E[u] \xrightarrow{t} E[u']} \quad (9)$$

3 Combinatorial Strand Algebra

Strand algebra is a process algebra [18] where the main components represent DNA strands, DNA gates, and their interactions. The basic algebra is non-deterministic algebra, and the further extension is a stochastic variant [10]. Strand algebras may look very similar to either chemical reactions, or Petri nets, or multiset-rewriting systems. The difference here is that the equivalent of, respectively, reactions, transitions, and rewrites, do not live outside the system, but rather are part of the system itself and are consumed by their own activity, reflecting their DNA implementation. A process algebra formulation is particularly appropriate for such an internal representation of active elements.

The Combinatorial Strand Algebra, P – basic strand algebra has some atomic elements (signals and gates), and only two combinators: parallel (concurrent) composition $P \mid Q$, and populations P^* . An inexhaustible population P^* has the property that $P^* = P \mid P^*$; that is, there is always one more P that can be taken from the population. The set P is formally the set of finite trees P generated by the syntax shown below; we freely use parentheses when representing these trees linearly as strings. Up to the algebraic equations described below, each P is a multiset, i.e., a solution. The signals x, y, \dots are taken from a countable set.

$$P ::= x; [x_1, \dots, x_n]. [y_1, \dots, y_m]; 0; P_1 \mid P_2; P^* n \geq 1, m \geq 0 \quad (10)$$

A gate is an operator from signals to signals: $[x_1, \dots, x_n]. [y_1, \dots, y_m]$ is a gate that binds signals x_1, \dots, x_n , produces signals y_1, \dots, y_m , and is consumed in the process. We say that this gate joins n signals and then forks m signals; some special cases are shown on the fig 4. An inert component is indicated by 0. Signals and gates can be combined into a 'soup' by parallel composition $P_1 \mid P_2$ (a commutative and associative operator, similar to chemical '+'), and can also be assembled into inexhaustible populations, P^* . Square brackets are omitted for single inputs or outputs.

Explanation of the Syntax and Abbreviations:

$$x \quad (11)$$

Signal

$$0 \quad (12)$$

Inert

$$x_1.x_2 \triangleq [x_1].[x_2] \quad (13)$$

transduser gate

$$P_1 | P_2 \quad (14)$$

Composition

$$x.[x_1, \dots, x_m] \triangleq [x].[x_1, \dots, x_m] \quad (15)$$

fork gate

$$P^* \quad (16)$$

Population

$$[x_1, \dots, x_m].x \triangleq [x_1, \dots, x_m].[x]$$

The relation $\equiv \subseteq P \times P$, called mixing, is the smallest relation satisfying the following properties; it is a substitutive equivalence relation axiomatizing a well-mixed solution[3] given lower:

$$P \equiv P \quad (17)$$

Equivalence

$$P \equiv Q \Rightarrow P | R \equiv Q | R \quad (18)$$

Congruence

$$P \equiv Q \Rightarrow Q \equiv P \quad (19)$$

$$P \equiv Q \Rightarrow P^* \equiv Q^* \quad (20)$$

$$P \equiv Q, Q \equiv R \Rightarrow P \equiv R \quad (21)$$

$$P^* \equiv P^* | P \quad (22)$$

population

$$P | 0 \equiv P \quad (23)$$

Diffusion

$$0^* \equiv 0 \quad (24)$$

$$P | Q \equiv Q | P \quad (25)$$

$$(P | Q)^* \equiv P^* | Q^* \quad (26)$$

$$P | (Q | R) \equiv (P | Q) | R \quad (27)$$

$$P^{**} \equiv P^* \quad (28)$$

The relation $\rightarrow \subseteq P \times P$, called reaction, is the smallest relation satisfying the following properties. In addition, \rightarrow^* , reaction sequence, is the symmetric and transitive closure of \rightarrow . Reaction is shown lower:

$$x_1 | \dots | x_n | [x_1, \dots, x_n]. [y_1, \dots, y_m] \rightarrow y_1 | \dots | y_m \text{ gate } n \geq 1, m \geq 0 \quad (29)$$

$$P \rightarrow Q \Rightarrow P | R \rightarrow Q | R \quad (30)$$

Dilution

$$P \equiv P', P' \rightarrow Q', Q' \equiv Q \Rightarrow P \rightarrow Q \quad (31)$$

well mixing

The first reaction (gate) forms the core of the semantics: the other rules allow reactions to happen in context. Note that the special case of the gate rule for $m = 0$ is $x_1 | \dots | x_n | [x_1, \dots, x_n]. [] \rightarrow 0$. And, in particular, $x. []$ annihilates an x signal. We can choose any association of operators in the formal gate rule: because of the associativity of parallel composition under \equiv the exact choice is not important. Since \rightarrow is a relation, reactions are in general nondeterministic.

Note that signals can interact with gates but signals cannot interact with signals, nor gates with gates. As we shall see, in the DNA implementation the input part of a gate is the Watson-Crick dual of the corresponding signal strand, so that the inputs are always 'negative' and the outputs are always 'positive'. This Watson-Crick duality need not be exposed in the syntax: it is implicit in the separation between signals and gates, so we use the same x_1 both for the 'positive' signal strand and for the complementary 'negative' gate input in a reaction like $x_1 | x_1.x_2 \rightarrow x_2$.

4 Insertion Machine for Combinatorial Strand Algebra

The representation of Petri nets as a composition of agents and environments was discussed in section 2.2. This will allow us to build a model driver based on the Petri nets. Consider a place-transition Petri Net with places x_i ; then, a transition with incoming arcs from places $x_1..x_n$, and outgoing arcs to places $y_1..y_m$ is represented in the combinatorial strand algebra as $([x_1..x_n].[y_1..y_m])^*$, where an unbounded population of gates ensures that the transition can fire repeatedly. The initial token marking x_1, \dots, x_k (a multiset of places) is represented as $x_1 | \dots | x_k$. Conversely, a signal in strand algebra can be represented as a marked place in a Petri net, and a gate $[x_1, \dots, x_n].[y_1..y_m]$ as a transition with an additional marked 'one-shot' place on the input that makes it fire only once; then, P^* can be represented by connecting the transitions of P to refresh the one-shot places (this was suggested by Cosimo Laneve). Therefore, the combinatorial strand algebra is equivalent to place-transition Petri nets, and can be easily implemented into the Insertion Modeling System, by using the model driver based on the Petri nets.

Conclusions

Strand algebras in general would allow the compilation of a high-level formalism into the DNA structures, using the methods advised by Cardelli in [10]. We have shown the way of implementation of the basic strand algebra – combinatorial strand algebra, in the Insertion Modeling System, by constructing the model driver for the system, based on the Petri nets. Combinatorial strand algebra deals with countable sets of signals/gates and so on (as well as Petri nets), we can extend it in future, to make it able to handle infinite sets, using the possibilities of insertion modeling, that works with infinite models. Implementation of combinatorial strand algebra to the insertion modeling system can be considered as the first step for building the insertion models of biological systems. However the further extensions of combinatorial strand algebra: Nested strand algebra [6], and Stochastic strand algebra, require a constructing of a probabilistic model driver, in order to implement them in the Insertion Modeling System. The stochastic semantics can be taken for example from the Stochastic Petri nets, which are just nets with rates on transitions and with an induced Continuous Time Markov Chain semantics.

References

1. Alexander Letichevsky, Olexander Letichevskiy, Vladimir Peschanenko, Igor Blinov and Dmitriy Klionov: (en) Insertion Modeling System And Constraint Programming. In: Ermolayev, V. et al. (eds.) Proc. 7-th Int. Conf. ICTERI 2011, Kherson, Ukraine, May 4-7, 2011, CEUR-WS.org/Vol-716, ISSN 1613-0073, pp. 51-64 (2011), <http://ceur-ws.org/Vol-716>
2. Gilbert D.R., Letichevsky A.A. : A universal interpreter for nondeterministic concurrent programming languages. In M. Gabbrielli (eds.), Fifth Compulog network area meeting on language design and semantic analysis methods (1996).
3. Letichevsky A.A., D.R. Gilbert: A general theory of action languages. *Cybernetics and System Analyses*, vol. 1, pp. 16--36 (1998).
4. Letichevsky A.A. and Gilbert D.R.: A Model for Interaction of Agents and Environments. In D. Bert, C. Choppy, P. Moses, (eds.). *Recent Trends in Algebraic Development Techniques*. LNCS, vol. 1827, pp.11--58. Springer (1999).
5. Letichevsky A.A.: Algebra of behavior transformations and its applications. In V.B.Kudryavtsev and I.G.Rosenberg (eds). *Structural theory of Automata, Semigroups, and Universal Algebra*, NATO Science Series II. Mathematics, Physics and Chemistry, vol. 207, pp. 241--272. Springer (2005).
6. Baranov S., Jervis C., Kotlyarov V., Letichevsky A., and Weigert T.: Leveraging UML to Deliver Correct Telecom Applications. In L. Lavagno, G. Martin, B.Selic (eds.). *UML for Real: Design of Embedded Real-Time Systems*. Kluwer Academic Publishers. Amsterdam (2003).
7. Letichevsky A., Kapitonova J., Letichevsky A. Jr., Volkov V., Baranov S., Kotlyarov V., Weigert T.: Basic Protocols, Message Sequence Charts, and the Verification of Requirements Specifications. *Computer Networks*, vol. 47, 662--675 (2005).
8. Kapitonova J., Letichevsky A., Volkov V., and Weigert T.: Validation of Embedded Systems. In R. Zurawski, (eds.). *The Embedded Systems Handbook*, CRC Press, Miami (2005).
9. Letichevsky A., Kapitonova J., Volkov V., Letichevsky A., jr., Baranov S., Kotlyarov V., Weigert T.: System Specification with Basic Protocols. *Cybernetics and System Analyses*, vol. 4, pp. 479--493 (2005).
10. Cardelli L.: Strand Algebras for DNA Computing (Preliminary version). *DNA Computing and Molecular Programming*, 15th International Conference, DNA 15. LNCS 5877:12-24, Springer (2009).
11. P. Yin, H. M.T. Choi, Calvert C.R., Pierce N.A.: Programming Biomolecular Selfassembly Pathways. *Nature*, 451:318-322 (2008).
12. Soloveichik D., Seelig G., Winfree E.: DNA as a Universal Substrate for Chemical Kinetics. *PNAS*, March 4, 2010, doi: 10.1073/pnas.0909380107.
13. Cardelli L.: On Process Rate Semantics. *Theoretical Computer Science* 391(3) 190-215,
14. Marathe A., Condon A.E., R.M.: Corn. On Combinatorial DNA Word Design. *J. Comp. Biology* 8(3), vol. pp. 201--219, (2001).
15. Milner R.: *Communicating and Mobile Systems: The π -Calculus*. Cambridge University Press (1999)
16. Danos V., Laneve C.: Formal molecular biology. *Theoretical Computer Science* 325(1) pp. 69-110 (2004)
17. Regev A., Panina E.M., Silverman W., Cardelli L., Shapiro E.: BioAmbients: An Abstraction for Biological Compartments. *Theoretical Computer Science*, vol. 325(1), pp. 141--167 (2004)
18. Cardelli L.: Artificial Biochemistry. In: A. Condon, D. Harel, J.N. Kok, A. Salomaa, E. Winfree (eds.). *Algorithmic Bioprocesses*. Springer (2009)

An Approach to Parallelizing Fortran Programs using Rewriting Rules Technique

Anatoliy Doroshenko¹ and Kostiantyn Zhreb¹

¹Institute of Software Systems of National Academy of Sciences of Ukraine,
Glushkov prosp. 40, 03187 Kyiv, Ukraine
doroshenkoanatoliy2@gmail.com, zhreb@gmail.com

Abstract. We present an ongoing research in the area of transforming existing sequential Fortran programs into their parallel equivalents. Our approach is to use rewriting rules technique in order to automate the transformation process. Sequential source code is transformed into parallel code for shared-memory systems, such as multicore processors. Parallelizing and optimizing transformations are formally described as rewriting rules which facilitates their reuse. Using high-level algebraic models allows describing program transformations in a more concise manner. Performance measurements demonstrate high efficiency of obtained parallel programs.

Keywords: rewriting rules technique, algebraic program models, multicore processors, Fortran, OpenMP.

Key Terms: High Performance Computing, Model, Methodology.

1 Introduction

Despite being one of the first programming languages, Fortran is still widely used, in particular for solving scientific and engineering computation-intensive problems. Its popularity is due to its relative simplicity and lack of complex facilities (e.g. pointers), closeness to mathematical description of problem and efficiency of generated binary code. Another reason for continued use of Fortran is that in more than 50 years of its existence a vast repository of programs, libraries and routines for solving different scientific problems has been created. Algorithms implemented in such programs are still valuable, however there is a need to adapt this legacy code to new parallel computational platforms. Furthermore, due to size and complexity of existing code, manual adaptation is not a practical option: there is a need for automated tools to facilitate conversion of legacy code to modern parallel platforms [5].

In this paper we describe an ongoing research on parallelizing Fortran programs using rewriting rules technique. Sequential source code is transformed into parallel code for shared-memory parallel platform (such as multicore processors) using

automated transformations. Parallelizing and optimizing transformations are formally described as rewriting rules which facilitates their reuse. Such approach is aimed at two main goals: to improve runtime efficiency of programs and to increase developer's productivity. We illustrate our approach on two sample programs: a simple Gauss elimination algorithm and an applied problem of calculating electron density from the field of quantum chemistry.

There has been an extensive research in the area of parallelizing existing sequential code, in particular for multicore architectures. Some approaches require manual code modification and provide facilities that help a developer express parallelism. Such approaches include parallel libraries [13], parallel extensions to existing languages [14] and new parallel languages [16]. Another research direction is interactive parallelization [11], when a developer manually selects the loops to be parallelized, and the tool applies transformation automatically (our approach also belongs to this category). Finally there are numerous approaches to automated parallelization, mostly implemented as parallelizing compilers [1]. Such systems use the static analysis of the source code to detect possible areas of parallelism and generate parallel binary code. Some papers also use the dynamic analysis to detect parallelism based on concrete input data [15], or machine learning approaches to select most appropriate transformations [17], or auto-tuning to discover optimal parameters of transformations [6]. The key differences of our approach is the use of the source-to-source transformations, allowing the developers to examine transformed program code, and the description of the transformations in terms of the formal models and rewriting rules, making easier for developers to add new parallelizing transformations or to modify existing ones.

This paper continues our research on automation of process of designing and development of efficient parallel programs, started in [3], [8], [9]. Our previous papers [3], [9] applied a similar approach to the development of parallel programs written in C# language for Microsoft .NET framework, while this paper concentrates on parallelizing Fortran programs. We have already described our first experiences with Fortran programs in [8]. However, as we moved from simple examples to real-world legacy code, we were forced to revise our approach, as described in this paper (see section 2). Also this paper places more significance on choosing place of application of existing program transformation, rather than developing new transformations.

Below we describe our approach in more detail, provide examples of parallelizing transformations and illustrate them with parallelization and evaluation of two programs: small example program (Gauss elimination) and applied quantum chemistry problem (electron density).

2 Our Approach: Algebraic Models and Rewriting Rules

As in our previous works [3], [8], [9], we use formal facilities, namely rewriting rules technique and high-level algebraic models of programs, to automate parallelizing existing sequential code. Legacy source code of sequential program written in Fortran is transformed into parallel version targeting shared-memory parallel platform

(multicore processors). As a part of transformation process, we create high-level algebraic models of legacy source code based on Glushkov algebra [2]. As described in [3], the models are created in two steps. First we use target language parser (Fortran in this paper) to build low-level syntax model, and then rewriting rules of special form (*patterns*) to extract language-independent algebraic operators from language constructs. Using high-level algebraic models allows describing program transformations in more concise manner. The additional benefit of such models when applied to legacy code is that they aid in understanding of algorithms by hiding the (frequently obsolete) implementation details. To this end, using multiple levels of algebraic models can be useful – e.g. the highest level describes just general structure of algorithm, while lower levels supply implementation details (the example of such models is described in section 3).

After high-level program model is created, we use parallelizing transformations to implement a parallel version of the program on a given platform. Transformations are represented as rewriting rules and therefore can be applied in automated manner. (Selection of loops that could be transformed is performed manually.) The declarative nature of rewriting rules technique simplifies adding new transformations. Also transformations work with high-level model elements (on any level of abstraction), which means they are language-independent.

Usage of high-level algebraic models also allows proving correctness of the developed transformations [3]. Based on program models, we have developed the algebra-dynamic models of program execution for multicore architecture using discrete dynamic (transitional) systems [2]. For these models, we have (manually) proved that each of proposed code transformations is correct under certain conditions, i.e. that initial and transformed programs are equivalent.

To automate program transformations we use the rewriting rules system Termware [7]. Termware is used to describe transformations of *terms*, i.e. expressions of form $f(t_1, \dots, t_n)$. Transformations are described as Termware *rules*, i.e. expressions of form `source [condition]-> destination [action]`.

Here *source* is a source term (a pattern for match), *condition* is a condition of rule application, *destination* is a transformed term, *action* is additional action that is performed when rule fires. Each of 4 components of a rule can contain variables (denoted as `$var`), so that rules are more generally applicable. Components *condition* and *action* are optional. They can execute any procedural code, in particular use the additional data on the program.

Termware supports a number of evaluation strategies, including TopDown (used in this paper), BottomUp and a possibility to implement additional strategies. Termware system itself doesn't check that transformation process terminates, however the rules used in this paper are designed in such way that each model element is processed at most once, therefore the transformation process is guaranteed to terminate.

In addition to rewriting system, our tools include parsers and generators for target languages that perform transformation between source code and low-level (syntax) program model, which is represented as Termware term. We have previously developed such tools for C# language [3], [9]; in this paper we have developed a Fortran parser and generator based on GCC Fortran Compiler.

3 Parallelization for Shared-memory Systems Using OpenMP

In this section we describe the process of parallelizing sequential Fortran programs for parallel systems with shared memory, such as multicore processors. We parallelize source code of Fortran programs by replacing suitable loops with parallel loop constructs. To create multithreaded Fortran program we use OpenMP framework [14]. OpenMP *PARALLEL DO* directives are used to parallelize loops. For simple loops, just addition of such directive can produce quite efficient parallel code. In this case there is additional advantage of keeping transformed parallel code similar to existing sequential code. In more complex cases (when there is data dependency between iterations) there is a need of more significant transformations, such as using OpenMP library subroutines for advanced thread management. In such cases, the transformed source code contains significant changes. However, usage of high-level algebraic models allows describing these changes in concise and understandable form.

We will describe the details of our approach using as an example a Fortran program implementing Gauss elimination algorithm for solving systems of linear algebraic equations. The Fortran source code was transformed into a low-level syntax model using developed parser, then into a high-level algebraic model using Termware patterns. When working with legacy code, we found it useful to apply several levels of patterns. First we used generic linear algebra patterns, such as vector and matrix operations. The obtained algebraic model was language-independent, but still quite detailed. Then we applied patterns specific to the problem in question. In this way we obtained schematic representation of algorithm useful for its understanding and deciding where parallelizing transformations should be applied.

The high-level model of relevant fragment of program has the following form:

```
DoCnt ( K, 1, N-1,
        FindMaxElement, CheckDetZero, SwapMaxRowColumn,
        CalculateRow(K), UpdateElements )
```

We will parallelize only two of the operators present in program, namely *FindMaxElement* and *UpdateElements*. Other operators have less computational complexity, therefore their parallelization is less effective.

Out of two operators, the simplest is *UpdateElements*, responsible for calculating new values for elements of submatrix:

```
UpdateElements = DoCnt ( I, K+1, N, Assign ( S, A ( I, K ) ),
                        DoCnt ( J, K, N+1, Update ( A ( I, J ), S ) ) )
```

Here, *DoCnt* denotes common DO loop with counter. The iterations of the outer loop are independent, so this fragment is easily parallelized. We use the following rewriting rule:

```
DoCnt ( $var, $start, $end, $body, _MARK_Parallel ) ->
ParallelDoCnt ( $var, $start, $end, $body )
```

The loop to be transformed is marked with *_MARK_Parallel* symbol to enable rule application. *ParallelDoCnt* operator is high-level model element responsible for parallel loop. In particular, for OpenMP platform it is transformed into *OmpParallelDo* operator that describes OpenMP directive represented in Fortran as a pair of special comments: *!\$OMP PARALLEL DO ... !\$OMP END PARALLEL DO*.

Notice that for C language the same operator is represented as a single pragma statement: *#pragma omp parallel for*. Therefore, using multiple levels of patterns allows us to provide operators that are common for given platforms, use these generic operators in most rewriting rules and then specialize them only when transforming program model back into source code.

While *UpdateElements* operator can be parallelized by simple application of OpenMP directive, the other operator *FindMaxElement* is more complex. It also has the form of loop, but iterations of the loop update the same set of variables (value of the maximum element in submatrix and its indices). This is the case of reduction, when some local values are calculated on each iteration and then merged into one global value. OpenMP supports such cases with REDUCTION clause, however only a set of predefined reduction operators are supported: while finding just maximum value can be accomplished using OpenMP directives, finding maximum value and indices where it occurs is not directly supported.

Therefore we need to provide transformations that parallelize the loop in general case of reduction. We represent *FindMaxElement* as following combination:

```
FindMaxElement=FindMaxElLoc1*...*FindMaxElLocTN*FindMaxEl
Reduct
```

On each thread we execute local version of operator (*FindMaxElLoc1, ..., FindMaxElementLocTN*), and then execute reduction operator *FindMaxElReduct* that combines local values into one global value.

Both already described parallelizing transformations are aimed at high-level structure of algorithm. However, as we observed in [3], low-level implementation details, in particular memory access, can have profound impact on overall performance.

In the Gaussian elimination program we have observed the same effect. We noticed that for certain sizes of input matrix ($N=256*M$) there was a sudden increase of execution time. We attribute this increase to the peculiarities of memory access: namely, caching adjacent matrix elements. For such matrix size, the adjacent matrix elements were put into the same cache items, therefore increasing the number of cache misses and greatly reducing overall performance. To overcome this peculiarity, we declare the matrix size as $N+1$ instead of N . The extra elements are not used in calculations, but they change location of elements and improve efficiency of memory access. The transformation is implemented with the following rules:

1. [Declaration(N,Integer,\$val):\$next]
 ->[Declaration(N,Integer,\$val):
 [Declaration(MN,Integer,\$val+MShift(\$val)): \$next]]
2. MShift(\$val) [\$val%32==0]->1 !->0

3. `Declaration(A,Array(Double,[N,N+1]))`
`-> Declaration(A,Array(Double,[MN,MN+1]))`
4. `Procedure($name,[N:[A:$next]])->`
`Procedure($name,[N:[MN:[A:$next]])`
5. `[Parameter(N,Integer,In):$next]`
`-> [Parameter(N,Integer,In):[Parameter(MN,Integer,In):`
`$next]]`
6. `Call($name,[N:[A:$next]])`
`-> Call($name,[N:[MN:[A:$next]])`

The rule 1 adds new parameter, MN, denoting declared matrix size. The rule 2 specifies for which values of matrix size the transformation should be applied. The rule 3 modifies matrix declaration to use new size MN instead of N. Rules 4-6 propagate new parameter to all procedures, procedure parameters and procedure calls.

Notice that rules 4-6 are applied multiple times in a single program: for each procedure definition (rules 4-5) and for each procedure call (rule 6). One of the advantages of rewriting rules technique is that single rule can describe changes in multiple places, reducing effort to make the changes and preventing mistakes possible when applying such changes manually. Notice also that rules 1-6 work on lower level of abstraction compared with previously described rules. The ability to describe transformations on different model levels is another advantage of proposed approach and it allows describing different types of transformations with the same tools.

4 Performance Evaluation: Test Program and Real-world Example

To evaluate effects of developed transformations, we have measured the performance of different versions of initial program of Gauss elimination. We have compared performance of 4 versions: initial sequential program (SEQ), parallel program with *UpdateElements* operator parallelized (PAR1), parallel program with both *UpdateElements* and *FindMaxElement* operators parallelized (PAR2), and program with both operators parallelized and memory optimization applied (MEM). The measurements were performed on 4-core parallel system, for matrix sizes from 256 to 2048. Obtained speedup (compared with SEQ program) is shown on fig. 1.

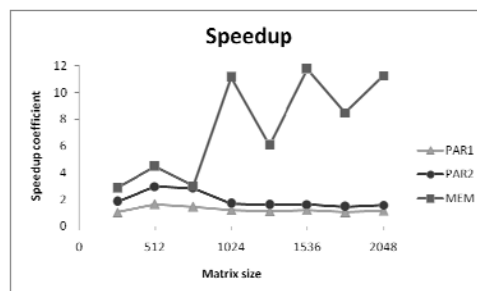


Fig. 1. Speedup of transformed programs (Gauss elimination).

As can be seen from the diagram, all transformations result in some performance increase, although their effect differs. For small matrix sizes, both PAR1 and PAR2 show some noticeable speedup, while MEM is not very effective and is very close to PAR2. However, for larger matrix sizes ($N > 1024$), the situation changes. PAR1 and PAR2 become less efficient, close to SEQ. However, MEM becomes much more efficient and demonstrates speedup of more than 10x. Therefore both high-level transformations of algorithms and taking care of low-level implementation details is necessary to obtain efficient parallel programs. Measurement results also demonstrate complex dependency of execution time on real parallel systems, as compared to ideal theoretical models that suggest simple $O(N^3)$ dependency.

After developing our tools on sample problem (Gauss elimination) we have tried them on real-world program in area of quantum chemistry. The program calculates electron and spin density in atoms of polycyclic aromatic hydrocarbons on $N \times N$ grid [12]. The size of the program is 1680 lines of Fortran code. Source code is not well structured – actual calculations are mixed with I/O operations, debug code and some hardcoded data. Also it contains mix of features from different versions of language – from Fortran 77 to Fortran 95. Therefore usage of high-level algebraic models helped us to understand this legacy code and apply parallelizing transformations in most efficient way.

We were able to reuse parallelizing transformations developed for Gauss elimination program also in electron density program. Only the first, most simple loop transformation was applied. However, the challenge was to select the most suitable loop for this transformation, as the program contained 54 loops and trying all of them was not a feasible option. We have used a profiler tool, Intel VTune Amplifier [10], to find hotspots in source code. Then we applied rewriting rules technique to detect all loops enclosing such code fragments. Thus the number of candidate loops was significantly reduced from 54 to 6. Out of these 6 loop, we applied transformation to second outermost loop (as the outermost loop contained too few iterations, and parallelizing inner loops was less efficient because of repeated cost of creating and synchronizing threads each time inner loop was executed).

We have compared execution time of initial sequential program (SEQ) and parallelized program (PAR) for grid dimensions N from 200 to 800 (see fig. 2).

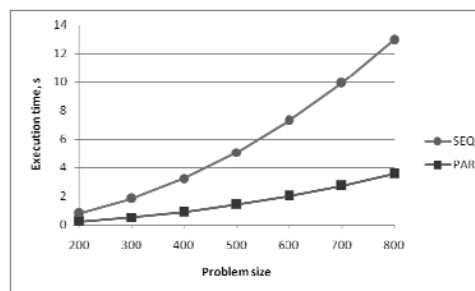


Fig. 2. Comparison of initial and transformed program (electron density).

Applying transformations has resulted in quite significant speedup – from 3.3X to 3.6X (depending on problem size) on 4-core system.

Conclusion

In this paper we have described our approach for parallelizing Fortran programs by applying formalized program transformations to existing sequential Fortran code. Using rewriting rules technique automates application of transformations and prevents mistakes that can appear when applying changes to source code manually. High-level algebraic models simplify understanding of legacy programs and their transformations, and enable transformation on different levels of abstraction. We have applied our approach both to simple program implementing Gauss elimination algorithm and real-world quantum chemistry problem (calculating electron density). Performance measurements demonstrate significant speedup for both programs.

Further research directions include development of the same approach for transforming legacy Fortran applications to target distributed-memory systems and GPUs. Our future plans also include extension to Grid and cloud-based platforms. Also we are planning to improve support for large and complex Fortran programs, in particular automate selection of most suitable place of application for transformations.

References

1. Allen, R., Kennedy, K.: *Optimizing Compilers for Modern Architectures: A Dependence-Based Approach*. Morgan Kaufmann, San Francisco (2001).
2. Andon, P.I., Doroshenko, A.Yu., Tseitlin, G.O., Yatsenko, O.A.: *Algebra-algorithmic models and methods of parallel programming* (in Russian). *Academperiodika*, Kiev (2007)
3. Andon, P., Doroshenko, A., Zhreb, K.: *Programming high-performance parallel computations: formal models and graphics processing units*. *Cybernetics and Systems Analysis* 47, 4, 659–668 (2011)
4. Asanovic, K. et al.: *A view of the parallel computing landscape*. *Commun. ACM* 52, 10, 56–67 (2009)
5. Buttari, A., et al.: *The impact of multicore on math software*. In: Kagstrom, B., Elmroth, E., Dongarra, J., Wasniewski, J. (eds.) *PARA 2006*. LNCS vol. 4699, pp. 1–10. Springer, Heidelberg (2007)
6. Datta, K., et al.: *Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures*. In: *ACM/IEEE Conference on Supercomputing (SC '08)*, pp.1–12. IEEE Press, Piscataway (2008)
7. Doroshenko, A., Shevchenko, R.: *A Rewriting Framework for Rule-Based Programming Dynamic Applications*. *Fundamenta Informaticae* 72, 1–3, 95–108 (2006)
8. Doroshenko, A.Yu., Zhreb, K.A., Tyrchak, Yu.M., Khatniuk, A.O.: *Creating Efficient Parallel Programs in Fortran Using Rewriting Rules Technique* (in Russian). In: *International Conference on High-Performance Computations (HPC-UA'2011)*, pp. 76–83. Kyiv, October 12-14, 2011
9. Doroshenko, A., Zhreb, K., Yatsenko, O.: *Formal Facilities for Designing Efficient GPU Programs*. In: *International Conference on Concurrency Specification and Programming (CS&P'2010)*, pp. 142–153. Bornicke, Sep. 27–29, 2010

10. Intel Parallel Studio <http://software.intel.com/en-us/articles/intel-parallel-studio/>
11. Ishihara, M., Honda, H., Sato, M.: Development and Implementation of an Interactive Parallelization Assistance Tool for OpenMP: iPat/OMP. *IEICE Transactions on Information and Systems* E89-D 2, 399–407 (2006)
12. Khavryutchenko, V.D., Tarasenko, Y.A., Strelko, V.V., Khavryuchenko, O.V., Lisnyak, V.V.: Quantum chemical study of polyaromatic hydrocarbons in high multiplicity states. *International Journal of Modern Physics B* 21, 26, 4507–4515 (2007)
13. Leijen, D., Schulte, W., Burckhardt, S.: The design of a task parallel library. In: 24th ACM SIGPLAN Conference on Object Oriented Programming Systems Languages and Applications (OOPSLA '09), pp. 227–242. ACM, New York (2009)
14. OpenMP specification. <http://openmp.org/wp/>
15. Rus, S., Pennings, M., Rauchwerger, L.: Sensitivity analysis for automatic parallelization on multi-cores. In: 21st Annual International Conference on Supercomputing (ICS '07), pp. 263–273. ACM, New York (2007)
16. Saraswat, V.A., Sarkar, V., von Praun, C.: X10: concurrent programming for modern architectures. In: 12th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '07), pp. 271. ACM, New York (2007)
17. Tournavitis, G., Wang, Z., Franke, B., O'Boyle, M.F.P.: Towards a holistic approach to auto-parallelization: integrating profile-driven parallelism detection and machine-learning based mapping. *SIGPLAN Not.* 44, 6, 177–187 (2009)

Conceptualization of University Structure as a Complex Mechanism Serving Educational Interests

Aleksander Spivakovsky¹, Lyudmila Alferova¹, and Eugene Alferov¹

¹ Kherson State University, 27, 40 Rokhiv Zhovtnya St., Kherson, 73000, Ukraine
spivakovsky@ksu.ks.ua, kuznetsova.mila@gmail.com,
alferov_jk@ksu.ks.ua

Abstract. This article focuses on university structure specification of conceptualization as a corporate complex mechanism which serves educational interests. Its exploration is needed for a more harmonious and balanced passing reformation process and modernization of education in Ukraine in accordance with European trends. Technical model of university was formed and comparison of the university and company structure was made. Some conclusions about existing problems of education process efficiency and the ways of its solutions were made. There are critical modern views on the education system as a competitive member of the economy market in this article.

Key words: modernization, labor market, reformation, competence, education process.

Key terms: CompetenceFormationProcess, LaborMarket, Qualification.

1 Actuality

Information society needs highly educated specialists who are effectively able to apply gained knowledge, prepared for democratic changes and social-oriented cooperation. In today's world, higher education is becoming more crucial to determine not only the level of education and culture of a particular person, but a whole society.

The main role in this process is assigned to higher education institutions. Therefore, the reform and modernization of higher education in Ukraine is really one of the actual problems on the way to innovation, the European development of Ukrainian society and enhancing the role of higher education in forming well-educated younger generation [1].

Success in achieving these goals involves the transformation of higher education in accordance with European requirements, defining the criteria which are:

1. quality in specialists preparation;
2. Fundamentalization and individualization of studying;
3. Strengthening of trust between the subjects of educational activity;

4. Compliance with European labor market;
5. Mobility of students and teachers;
6. Compatibility of qualifications;
7. Strengthening the competitiveness of university graduates, etc.

It should be noted that a positive result from any changes can be achieved only with full understanding of internal relations and the fundamental understanding of complex system model of higher education.

There are many approaches to the definition of education. One of them is: Education is a part of material life production, which with the help of two connected training and education processes provides purposeful formation necessary for the society type of person. University education is the main component of higher education [2].

This paper describes one of the explicit models of structuring the university as a corporation. Higher education institution is a center for the formation of leading specialists that is why the main factor is human resources. In addition, along with technological bases of production in higher education is the most important question of harmonious synthesis of pedagogical and psychological concepts, deep psychological understanding of patterns of learning activities, principles and methods of teaching and learning process.

2 Features of the Educational Process at the Higher School. Aims of Professional Training

Aims of professional education perform system creation function in educational activities. The choice of content, methods and means of learning and education depends on choosing the goals.

There are a lot of educational goals types. You can select a state legislative aims of education, community goals, initiative goals of teachers.

Normative state goals – are the most common goals identified in government documents and state education standards. At the same time there are social goals – goals of different social classes that reflect their needs, interests and demands of training. For example, goals of the employer belong to specific goals. Teachers take these conditions into account by creating different types of specializations, different concepts of learning. Initiative goals – are the goals developed by practicing teachers and their students including type of school, profile of specialization and academic subject, considering the knowledge level of students and teachers preparation.

The set of final goals is a list of tasks that should be able to decide at the end of specialist training. These are called models of specialist.

According to N.F. Talyzina the first transition step from model of specialist to model of his professional training is the selection and complete description of typical tasks that he will must solve in his future professional activity. Typical tasks are organized in a hierarchy, which is also the hierarchy of higher education goals.

1. The top positions in this hierarchy are the tasks that should be solved by all professionals, regardless of the particular profession or country of residence. They

are determined by the character of the historical epoch and can be called like tasks of the century.

2. The specific tasks in the country form the second level. Now in our country the problems connected with the development of market relations and problems of international relations are especially important.
3. The third level – professional tasks, are the largest by volume and variety of solvable problems.

Based on the analysis of all tasks types and on exclusion of repeating elements the model of a specialist activity can be made. Which in its turn is the basis for the constructing a model of the university structure.

3 The Paradigm of Technical Model of the University

Professional skills are the most important graduate's characteristics. While focusing just on this aspect let's try to construct a technical model of the university activity.

The paradigm of technical models of the university lies in the similarity of economic corporation structure of production with the system of higher education.

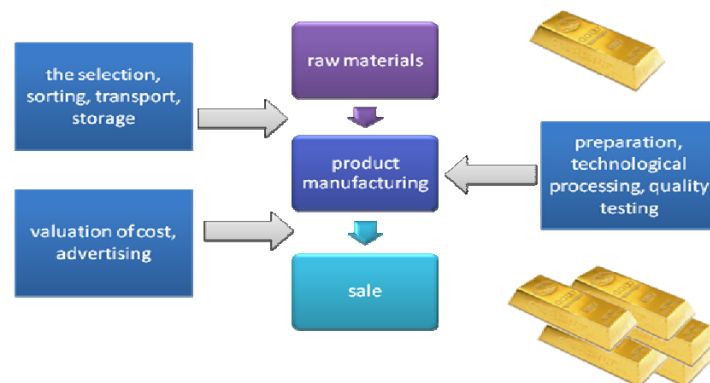


Fig. 1. General scheme of enterprise manufacture.

Corporation is now the dominant form of business. As a large and influential organization, they are usually attractive to the society. They have a social orientation, social responsibility, comply with many rules and laws, consumer rights and interests of workers and whole society. Great value for the corporation has a corporate culture, established rules of behavior of employees, traditions, ethics.

Today, in a market economy should be represented a higher educational institution as a corporation, serving the educational interests of the state.

Above is shown the general scheme of any enterprise. It has three key levels. Each of them under the influence of organic factors certain changes occur and increased

cost of manufactured products. There are many ways to increase the efficiency for corporation.

Let us consider now the scheme of competitive specialist training in high school. Like the previous, it consists of three levels.

The first level is a graduate student. The set of graduates can be considered as external resources. Entrants have their own skills, knowledge and abilities. But at the labor market they have almost zero cost. This is due to a low level of training and lack of a clear relationship between: I know \rightarrow I can \rightarrow I do.

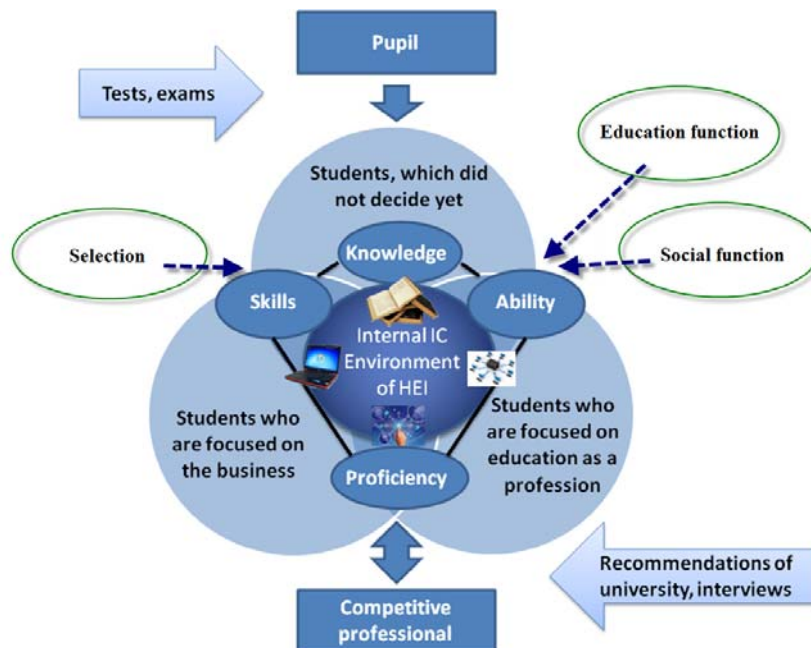


Fig. 2. Preparation of the leading specialist in high school (Internal IC of HEI – internal information and communication environment of higher education institution).

At the second level under the influence of training and educational process is phased synchronous integration of four key rods: knowledge, proficiencies, skills and abilities [3].

So at this stage there is a question of efficiency of the learning process. Let's make an analogy with company and build possible examples of implementation concerned with high schools (see table below).

Depending on the personality of the students and the level of internal motivation Maryeyev D.A. in his article "Understanding the psychological factors of successful students" identifies three categories:

1. Students who are focused on education as a profession. The most important is their interest in future work, the desire to realize themselves in this way. They observed a tendency to continue their education in PhD programs.

2. Students who are focused on the business. Education for them is a tool (or start step) to open their own business, trade, etc. They understand that with time this area will require education, but to his attitude in the profession they are less interested than the first group.
3. Students who, on the one hand, can be called "those who are undecided" on the other – suppressed issues of personal, domestic nature. Most of them can't graduate successfully.

Table 1. Realization of methods for improving efficiency of enterprise in high schools.

Methods of improving the enterprise efficiency	The implementation of method in the higher educational institutions
Increase of income	Bringing scientific and technological capabilities and resources for the national and international financed projects implementation.
Increasing of product quality	Publication of information resources on the portal of the University, its constant updating, on-line communication between students and teachers. To provide students with access to educational information resources for the improvement of quality individual work.
Improving of staff competencies	Using an integrated, corporate, personalized information and analytical system of business processes of the University. Opening e-mail service for teachers and students, a quick access to necessary information resources of other schools.
Equipment modernization	Modernization of scientific laboratories equipment. Organization network access, including through radio-access point, in academic buildings and hostels, both to external and internal information resources teachers and students.
Technology optimization	Through the implementation of student-oriented and distance learning technologies.

The first two categories of students under the influence of internal information and communication environment of universities are highly skilled graduates who are active in the labor market demand. That is, their cost considerably increases.

The modern system of higher education, except educational and social functions also plays a selective role. Other words selects the most talented part of graduates in higher education, gives them a chance to realize their creative ambitions and ultimately form the elite of society, really need the labor market. For example, in the University of Glasgow (Caledonia) learning in a 1st course dedicated to actual equalization, adaptation and selection of students who will receive bachelor qualifications on the next courses. This university proud of the fact that in the last 20 years his contingent has increased significantly – from 36 to 52 percent of graduates. Thus, here is a problem of heterogeneity of first course contingent. However, focusing on the social component for the students of the 1st year, the administration not only solves the problems of adaptation, selection and equalization, providing appropriate conditions and in the process includes a significant number of the best school teachers, but also creating a strong motivation of educational activity of their new students. This approach provides the possibility of building conscious of their own learning path for the undergraduate. In Ukraine this approach, unfortunately only in the first semester, Kiev-Mohyla Academy realizes.

The behavior and professional activities of each person affect certain individual motives and values. So employees must be interested just in time, and by analogy with students, must search for the solution of tasks aimed at achieving the essential interests of the company. Understand the structure of motives an employee the company may find leverage of influence. For example, one of the main tasks of the university is to achieve high indicators in professional graduates and their subsequent successful employment. Created conditions for achieving this goal can qualitatively supplement staffing of the university.

Having considered the model of university technical side, having an analogy with the corporation and highlighting the basic regularities and principles of enterprise efficiency improving we can achieve positive results in increasing demand for graduates.

4 Conclusions

In the Fig. 3 you can see the one model of relationships harmonization between HEI and labor market.

If we will make a comparative description of each level of material production in enterprise system and system of getting education on the example of high school, the following conclusions will take place.

University – a complex multi-functioning production of intellectual product.

The main goal of higher education is to increase the professional value and cost of graduates in the labor market.

The success of harmonious integration of entities that can fundamentally change human priorities and values depends on the level of organization and technical equipment and on internal information and communication teaching environment.

Motivation of students to learning creates conditions for further elaboration and improvement of high education establishment and university scientific level.

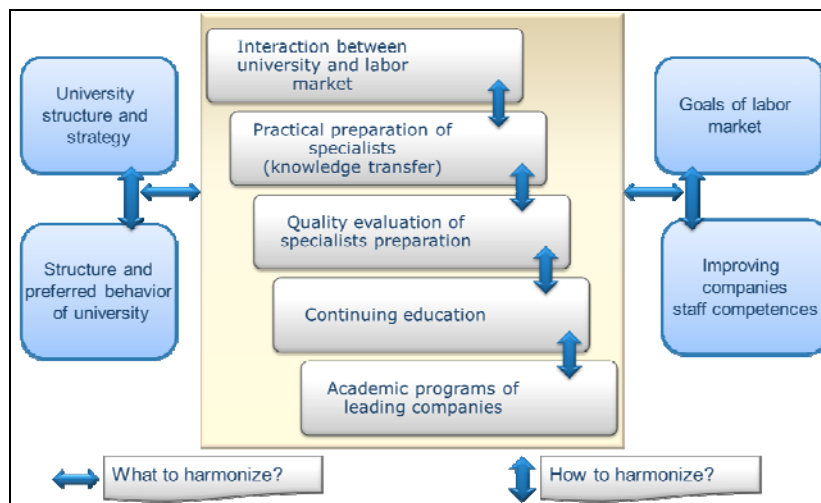


Fig. 3. Harmonization model.

References

1. Conceptual foundations and directions of development higher education of Ukraine: by the state on October 14, 2011. Ministry of Justice of Ukraine. Official publishing. Parliament publishing, Kyiv (2011) (in Ukrainian).
2. Goncharenko, S.U.: Fundamentalization of education as didactic principle. Shlyah osvitu, vol. 1, pp. 2--6 (2008) (in Ukrainian).
3. Petuhova, L.E., Spivakovsky, O.V.: The main problems of modern higher education didactics. Computer in school and family. vol. 3 (91), pp .13--15 (2011) (in Ukrainian).

On the Problem of Multi-Channel Communication

Michal Nagy¹

¹ Faculty of Information Technology, University of Jyväskylä,
P.O.Box 35 (Agora), FI-40014, Jyväskylä, Finland
michal.nagy@jyu.fi

Abstract. This paper discusses the issue of multi-channel communication. Multi-channel communication is a vision of future electronic business communication that is based on abstract messages, formal domain descriptions, formal communication channel descriptions and context-aware computing. We present research questions related to this vision together with a solution in form of a self-managed communication service. We propose the use of ontology-based approach for modeling and utility functions for decision making. Such a service can be proactive, capable of learning and improving itself.

Keywords: communication, ontology, adaptive system, context-aware system, B2C

Key Terms: InformationTechnology, BusinessIntelligence

1 Introduction and Motivation

Nowadays, businesses communicate using various electronic communication channels such as text messages, e-mails, newsletters, etc. The communication happens in both directions – inbound and outbound. It is difficult to keep track of messages on different media and in both directions at the same time. Since every communication channel is different in its nature, each one requires a different method of message composition. Therefore the channel has to be known before the message is written.

In this paper we are trying to identify problems related to business-to-customer (B2C) multi-channel communication and propose a set of techniques that could solve these problems. In section 2 we define multi-channel communication and terms related to it. Section 3 addresses problems of multi-channel communication and raises several research questions. In section 4 we propose an approach based on a self-managed system. Finally, section 5 discusses the future work and concludes the paper.

2 Multi-channel Communication

Human-to-human electronic communication can be understood as a process of information exchange between two or more parties. Information is usually exchanged in a form of messages. These messages can be e-mails, spoken sentences, text messages, blogs, etc. In general, communication can follow many patterns and have different requirements on the communicating parties. We concentrate on business communication, mainly on business-to-customer communication.

Let's imagine that an Internet store wants to advertise a new computer model. Therefore the store composes a newsletter and sends it to its customers. One of the customers is interested in the computer and asks a question about this product by e-mail. The shop also answers by e-mail. After a week the customer decides that he wants to buy the computer and makes an order using a form on the Web page of the store. The store confirms the order by an email. This part of the communication involved several communication channels – newsletter, e-mail and a Web form. If we are looking at this communication from the store's perspective, all these messages are interconnected, even though they originated from different communication channels and at different times. A system that can seamlessly integrate these messages would detect a new pattern that is valuable for the Internet store. A system that could also target the right customers using the right channels would be even more beneficial. In order to understand such a system we have to understand its components.

Communication happens through media. In general the definition of the medium depends on the abstraction level. On the level of signaling, medium can be understood as the surroundings that are used to transfer the signal. An example would be copper wire, optical fiber, air, etc. On a higher level, medium can be considered to be a method of communication that follows certain high-level network protocols. An example would be a computer network, telephone network, etc. Let us call this higher level medium a logical medium. One logical medium can operate on several physical media – e.g. a computer network can operate on copper wires, fiber optics or wirelessly.

One logical medium can host different types of communication channels. Under a communication channel we understand a set of communication methods that cover a way of business-customer communication. An example of such a channel could be e-mail channel, telephone channel, text message channel, instant messaging channel, etc. We introduce this term as an abstraction from the low level communication method. The relationship between media and communication channels is shown in Figure 1.

Each communication channel has its advantages and disadvantages in terms of speed, clarity, maximum size of the message transferred, cost, etc. Therefore it is reasonable to assume that each channel is suitable for a different set of circumstances. There are several things that are relevant to the interaction between the user and the system. Information about these things forms the context [1]. Therefore, in order to find the best channel, we need to make the system context-aware.

Based on the direction of communication we identify two main types of communication channels – simplex and duplex. FS-1037C defines simplex circuit as “a circuit that provides transmission in one direction only” [2]. Similarly, simplex communication channel is a channel where the communication is possible in one

direction only. Duplex communication channel is a channel which both parties can use to communicate with each other. A system using duplex communication has to be able to analyze communication in both directions. This might be a problem, because whereas the meaning of an outbound message is clear, an inbound message received by the system has to be analyzed first.

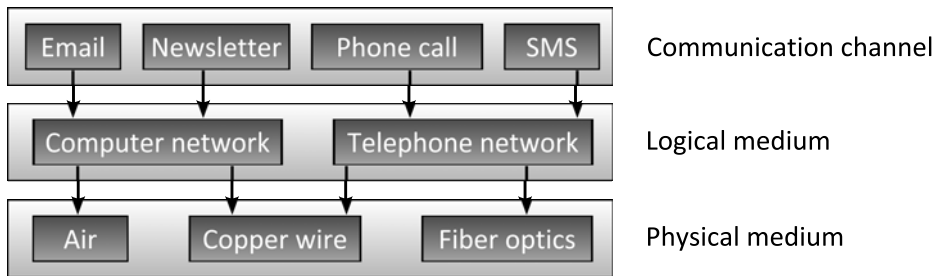


Fig. 4. The relationship between media and communication channels

3 Issues Related to Multi-channel Communication

Nowadays, we have many services that allow us to communicate using a variety of communication channels. Every communication channel has its limitations and those have to be taken into consideration when writing a message. Therefore it is necessary to know which communication channel will be used before the message can be composed. Also, if several communication channels are used, it is difficult to integrate them into one communication thread due to their heterogeneity.

What we are lacking is a system that would seamlessly integrate several communication channels and allow us to think of communication in more abstract terms. A system like this could choose the best possible communication channel based on the communication context and an abstract message. Rather than containing concrete text, an abstract message could contain a set of goals. In such a case, the knowledge of the communication channel would not be required in time of message composition. The system would also be able to integrate a reply, even though it was received from another channel. Therefore the communication would be transparent to the user of the system. We call this the vision of autonomous multi-channel communication. Figure 2 explains the vision in more detail.

Before this vision can become reality, several issues must be solved. Firstly, there has to be a way to model the abstract message and communication context. A modeling method like this should offer sufficient flexibility and expressiveness, but on the other hand keep good computational properties. Secondly, an abstract message has to be translated into a concrete message by taking the communication channel into account. This issue is closely related to the first one. Thirdly, the system has to be able to extract information elements from a received message. This would vary based on channel used. In some channels, the form of the message can be controlled by the system (e.g. a Web form). In other channels a deeper analysis is needed (e.g. e-mail

written by a human). Moreover, there is the issue of the communication channel selection. Knowing the communication context and abstract message definition, which channel is the best to use? Lastly, we assume that every communication is happening for a reason. That means it has a purpose/goal. This goal is related to a certain domain. If we want to formally describe a message, then there is a need to have a formal description of the domain the message is about. Also, a formal description of the communication itself is required. How can we model these domains so that they are compatible with other elements of the system?

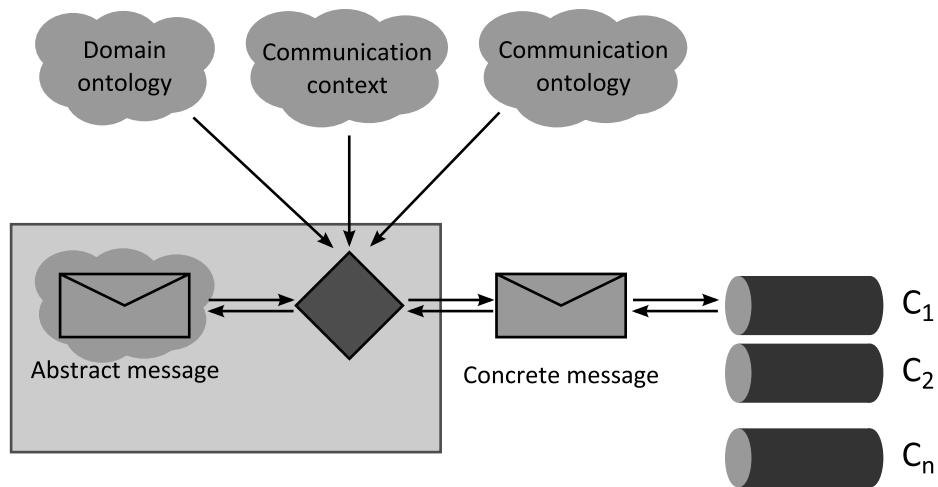


Fig. 5. Vision of autonomous multi-channel communication

4 Self-Managed Multi-channel Communication Service

Our proposed solution consists of a cloud communication service that would enable transparent multi-channel communication.

Strang and Linnhoff-Popien analyzed several context-modeling approaches based on six criteria [3]. We believe that the most relevant criteria are the following. Firstly, it is the ability to partially validate context information against a context model. This property is important for proper decision making. The second property is the ability to deal with incomplete and ambiguous information. This property is important during the examination and information elements extraction of the received message. These elements might be ambiguous, due to the nature of human language. The third criterion is the level of formality. A high level of formality is a way to avoid misinterpretations of the data. Based on these criteria, the most suitable way to model the context is an ontology-based approach.

In case of information elements extraction from a received message, some information channels are inherently easier to work with than others. Among information elements, there are two important ones that can greatly reduce the context

– the sender of the message and the topic of the communication. In case of electronic communication the sender can be determined based on the source address (e.g. e-mail address, phone number, etc.). The topic of the communication can be determined by using reference numbers, which is a well-known technique.

We suggest utility functions as methods of channel selection. According to Kephart and Das they are superior to action rules and pure goal policies, due to their ability to assign a numeric value representing the desirability of a future state [4]. The future state can be represented as a state when a particular message is sent to a particular channel under a particular context. Therefore the abstract message, communication context and channel description determine the inputs of the utility function.

A possible solution to the problem of abstract message transformation is a template-based approach. There will be a set of message templates and each template will have a formal specification of an abstract message that it needs in order to generate a concrete message. Similarly to channel selection approach, we can use utility functions to choose the proper template based on the communication context, abstract message and the template description.

Closely related to context modeling is domain modeling. We suggest ontology-based modeling as in the previous case for the compatibility reasons.

We believe that it would be of great benefit to design this system as a self-managed system. Self-managed system is a system which can maintain itself without human's intervention [5]. Cheng et al. believe that self-management can be achieved using closed-loop control based on reflection [6]. We believe that this solution might be feasible due to the fact that many components of the system would be formally described (e.g. template rules, utility functions, etc.) and thus easy to reason about. Such a system could optimize itself by modifying these components.

5 Conclusion and Future Work

In this paper we presented an issue related to the area of human-to-human business communication using electronic media. We present a vision that would allow businesses to integrate all the communication from and to their customers using any electronic communication channel. We call it the vision of multi-channel communication.

There are several issues related to this vision. We believe that there are a few important ones. Firstly, it is the issue of abstract message modeling and communication context modeling. Secondly, there is a problem related to message transformation from its abstract form to its concrete form. Thirdly, a way has to be found that would allow extraction of information elements from inbound messages. Fourth of all, we need a method of finding the best communication channel based on the communication context and the abstract message being sent. Lastly, it is the question related to the model of the communication domain.

We also provided a short description of a service that could solve these issues. We suggested an ontology-based approach to the problem of domain modeling, communication context and abstract messages. In case of message conversion we suggested a template-based approach, where each template would formally describe

what messages it can consume. We believe that the problem of inbound message element extraction can be solved by reducing the context and thus decreasing the ambiguity of the data. The problem of suitable communication channel selection can be solved by the use of utility functions. Finally, we proposed an ontology-based approach for the domain description as well.

In the future we plan to develop an ontology reflecting the domain of communication. At this point it appears that the use of description logic is the most appropriate way, since it has been well studied and the market offers high quality tools. We also would like to look at the area of context modeling using description logic. This is closely related to utility functions.

Acknowledgements. The author would like to thank the industrial partner IPSS (Intelligent Precision Solutions and Services) for bringing this real-life problem to our attention and for useful discussions. Further thanks go to the members of IOG (Industrial Ontologies Group) for their comments during the initial stages of this paper. Finally, the author would like to thank the TIVIT Cloud Software Program and COMAS graduate school for supporting this work.

References

1. Dey, A.K.: Understanding and Using Context. In: Personal Ubiquitous Computing, vol. 5, pp. 4--7. Springer, London (2001)
2. Federal Standard 1037C, http://www.its.bldrdoc.gov/fs-1037/dir-033/_4894.htm
3. Strang, T., Linnhoff-Popien C.: A Context Modeling Survey, In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham (2004)
4. Kephart, J., Das, R.: Achieving Self-Management via Utility Functions, In: Internet Computing, vol. 11, pp. 40--48. IEEE Computer Society (2007)
5. Kephart, J.O., Chess, D.M.: The Vision of Autonomic Computing. In: Computer, vol. 36, pp. 41--50. IEEE Computer Society (2003)
6. Cheng, S., Garlan, D., Schmerl, B.R.: Making Self-Adaptation an Engineering Reality, In: Self-star Properties in Complex Information Systems, Lecture Notes in Computer Science, vol. 3460, pp. 158--173. Springer (2005)

KSU Feedback Service as a Tool for Getting Feedback in Educational Institutions. Perspectives of Use in Government Organizations and Commercial Companies

Dmitry Kutetsky¹ and Valentina Gritsyuk¹

¹ Kherson State University,
dmkutetsky@gmail.com, gritsyuk@ksu.ks.ua

Abstract. The paper relates with current and potential using of KSU Feedback service. KSU Feedback allows users to get anonymous feedback from limited group of respondents. Article is compared this service with already existing in the Web similar services and describes potential ability of use KSU Feedback in educational, government organizations and commercial companies how tools to get feedback. Also described set of scenarios of use this service and proposed a number of modifications, which are allows to efficiently use KSU Feedback in specified areas.

Keywords. Feedback, educational process, interrogations, anonymous surveys, software.

Key Terms. ICTTool, Competence, Technology

1 Introduction

1.1 KSU Feedback

The developers have positioned their product as "Tools for anonymous or normal voting on clearly defined criteria for strictly defined set of respondents" (<http://feedback.ksu.ks.ua/>). The main purpose of the service is an anonymous vote and the normal vote (without anonymity i.e. when the estimation is linked to a specific person) only by means of service.

1.2 Short Description of Service

At the current moment there is a huge number of evaluation systems and feedback (we'll speak about them further). KSU Feedback is designed as a tool for feedback student-teacher for usage in high schools and contains some important features that traditional feedback systems do not have [1]:

- students anonymously evaluate the quality of knowledge provided by teacher;
- a teacher sees students' opinions about advantages and disadvantages of his system of teaching.

Accordingly, KSU Feedback has the following characteristics:

- the anonymity of respondents;
- conducting of survey among a certain number of respondents;
- limited access to the polls.

Next, we consider each of these features separately.

1.3 How does it Work / Principle of Functioning

Currently, standard operating procedure of the evaluation of teacher in KSU is as follows:

- The administrator of the resource or the person responsible for conducting the vote, creates a profile with a list of questions for respondents.
- Creates a survey which shows form (or many forms) for this survey and the number of respondents.
- Service generates the appropriate number of unique keys for voting and provides a list of these keys to the specified mailbox. Keys are printed and provided to a group of students to vote.
- Keys are given for interested students (respondents) at random order, then the students click (or type) the links of KSU Feedback service url for voting, enter their key and vote (at this stage, the student can choose survey he wants, or to view the overall results of passed surveys in the form of diagrams).
- Students can vote at the university as well as at home, or anywhere else with access to the Internet.
- After the expiration of the voting person who conducted the survey provides access to survey results to the teacher.

An example of a diagram(Fig.1) is the result of the work of service (a diagram is in the public domain due the permission of the owner and is available at <http://feedback.ksu.ks.ua/predef/reports/quick/966/>)

Note: The results of survey are available only to voters.

1.4 Key Features of the Service

- Anonymity - the keys for voting are unique and generated randomly, without reference to a specific person. Any mechanisms that help to identify the respondents are not used during voting process. The respondent can also see the

overall results of the surveys that he or she has already passed by using the same key for authentication.

- A flexible system of rights that includes:



Fig.1. Example of voting results presented by chart.

- Editing of the general rights, the rights of specific users and the rights of user groups.
 - Inheritance of rights.
 - Exposure of the different rights of access for a folder, the survey - reading, reading — writing, public access (survey is available without authorization).
- Voting among the limited number of respondents by setting the number of generated keys and providing the keys only for a certain range of respondents.
 - Graphical presentation of the results [2], [3] of survey (Fig. 2 shows an example), the presentation of summary reports on the general diagram (Fig. 3 shows an example).



Fig.2. Graphical presentation of the survey results.

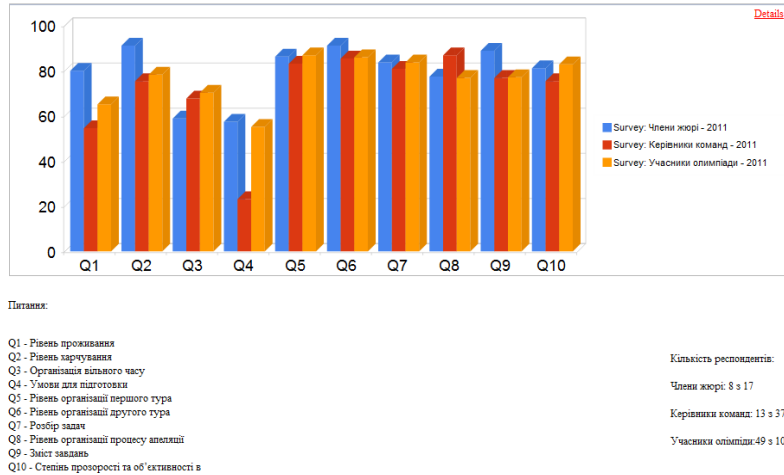


Fig.3. Summary report on the overall chart.

- Availability of API (Application Programming Interface) is a set of functions provided by KSU Feedback for use in exterior applications. Currently API allows the following:
 - Getting the voting results about a particular survey
 - Obtaining of all profiles that are linked to the survey
 - Embedding of a widget of report to a third-party site (using the page of designers of the widgets on the service - <http://feedback.ksu.ks.ua/wiki/widgets/>)

2 Comparison with Existing Services of Surveys. Common and Distinctive Characteristics

1. Vashopros.ru (<http://www.vashopros.ru/>)

Common features:

- Questions of several kinds (choice of one correct answer from a number of several answers, multiple choice, choice from the list).
- Fixing of permissions to view the voting results.

Differences:

- The ability to place HTML-widget for voting on third-party site - positive feature of this service.
- Ability to configure the visual design of the survey results: indicators, color, sound, etc.
- Adding of comments to the survey (limited to 100 comments) - positive feature of this service.
- Automatic deletion of the survey if it has not been used for 3 months - positive feature of this service.

2. Pollservice.ru (<http://pollservice.ru/>)

Common features:

- Number of created surveys is not limited.
- Multiple Choice. Respondent can choose several answers in surveys with multiple choice.
- Preview of the survey during its creation.
- Placement of link of survey in a blog, forum and social network.

Differences:

- The number of answers on the survey is not limited - negative feature of this service.
- Creation of HTML-poll widget to place on the site - positive feature of this service.
- Free answer. In a survey of free-answer respondent can enter his own answer, if none of the proposed answers is not fit him - positive feature of this service.
- Range of the respondents are not limited by means of the service (voting is free, but with protection from re-election by the tracking of IP-address) - negative feature of this service.

3. Whatiswrongwith.me (<http://whatiswrongwith.me>) - free service to get feedback.

Differences:

- Feedback supports several languages - positive feature of this service.
- The form of Feedback is arbitrary text - negative feature of this service.
- Review (comment) is published on behalf of Anonymous or your real name (by the choice of the respondent) - positive/negative (depends on the purpose of the survey) feature of this service.
- The number of respondents is not limited - negative feature of this service.
- The result of the survey is a list of text reviews - negative feature of this service.
- The results of the survey always are available to all (public) - negative feature of this service.

4. Prepod.org (<http://www.prepod.org>)

Common features:

- Global surveys (choice of one from several options, multiple choice).
- The anonymity of the respondent.
- View of the survey results is possible after the answering.

Differences:

- Ability to write a comment about a particular teacher - positive feature of this service.
- Range of respondents is not limited - negative feature of this service.

5. Sneak (<http://yabeda.net/prepodavатели/>)

Common features:

- View of the results of the survey is possible only after answering the question.

Differences:

- Comment is written in free text form - negative feature of this service.
- You can write a review on this site without a key / password / recording (anonymously), i.e. range of respondents is not restricted - negative feature of this service.

- Has various possibilities for creating events, forums, blogs with video and photo materials - positive feature of this service.
- 6. Critizise (<http://www.critizise.me/>)
 - Common features:
 - Ability to provide access to the respondents to the survey.
 - Differences:
 - Range of respondents is not limited. The voting is carried out by click on the link without a key - negative feature of this service.
 - Anonymity is not supported. Friends and colleagues participate in the survey, they receive the link through twitter or facebook - negative feature of this service.
 - Ability to determine the feedback form (the choice of one option of several, the answer in a text form, the answer in the form of a rating) - positive feature of this service.
 - Integration with social networks - positive feature of this service.
- 7. Vzyatochnik (<http://vzyatochnik.info/>)
 - Common features:
 - Anonymity is supported.
 - Differences:
 - Range of respondents is not limited. There is no protection against re-voting (the same person can express his/her opinion unlimited number of times changing his/her name) - negative feature of this service.
 - You can view the rating of feedback (complaints) - positive feature of this service.
 - Ability to receive feedback in the free form text (stories about the unfair treatment of students in high schools, about the excess of power by the teachers or the administration of universities) - positive feature of this service.
 - Integration with social networks - positive feature of this service.

The results of comparison of KSU Feedback with existing services of survey in the network are follows:

There are many services of surveys, and we gave a brief description of some of them (generally the work of around 20 services was analyzed). Other services that we analyzed are follows:

- Feedback system of the University of Sydney (<http://www.itl.usyd.edu.au/FEEDBACK/orderSF.cfm>).
- QuestionPro (<http://www.questionpro.com>).
- SurveyMonkey (<https://ru.surveymonkey.com/>).
- KwikSurveys (<http://kwiksurveys.com/>).
- BigPulse (<https://www.bigpulse.com/mpm/signin?url=%2Fmpm%2F>).
- Vovici (<http://www.vovici.com/>).
- Zoomerang (<http://www.zoomerang.com/>).
- CheckBox (<http://www.checkbox.com/>).
- Free Online Surveys (<http://freeonlinesurveys.com/>)
- Inquisite (<http://www.inquisite.com/>)
- Survey.io (<http://survey.io>)
- Feedaback.com (<http://feedback.com>)

- Concept feedback (<http://www.conceptfeedback.com/>).
- User voice (<http://www.uservoice.com/>).
- Poll Daddy (<http://polldaddy.com/>).

We analyzed all of the services for interviews, as web search did not reveal the full analog of KSU Feedback. KSU Feedback Service is the only service of the considered, which has a number of the following properties:

- The anonymity of the respondents (key vote).
- A certain number of participants in the survey (key vote).
- Limited access to information (access rights) [1].

The combination of these properties allows to speak about the possibility of using this service as a tool for getting anonymous feedback from a limited number of respondents.

KSU Feedback is created and used currently in the Kherson State University as a tool for get feedback in student-teacher relationship; we also consider other possible areas of application. But even the use of tools in the environment for which it was created has variations in the methods and purposes of use (variants of use - the purpose of the survey):

- The student does not see the results of the survey, the survey results are not available publicly - the teacher gets the feedback and has stimula to development and with the help of the graphical representation of data identifies the points that should be improved by students' opinion (punctuality, knowledge of the subject, use of modern technologies, the availability of material, etc.).
- The student sees the results of the survey -he can compare his evaluation of teaching the subject with an overall assessment.
- Results of the survey are publicly available - this is a kind of "rating" which is made by students about how subjects are teaching at the university.

It is often a very important factor for students that they can influence the learning process and convey their views to the teacher, and this ability is allow the respondent (student) can provide objective information due to the anonymity [4], [5].

In addition, there are several points that need to be taken into account when using KSU Feedback as a tool for get feedback between student and teacher, or between two other parties, where the respondent's anonymity is a major factor (student-teacher, the junior-chief, etc.) [6], [7]:

- The issue of anonymity. As we have pointed out, in some cases, the issue of anonymity for the respondent is the most fundamental. Distrust of the anonymity of the respondents is quite clear - unknown service selected by the leadership, lack of transparency of the work (of the user), lack of understanding of the principles of the network, etc. Therefore, the main factors for receiving objective results during conducting surveys are follows: explanation of the principles of service, voluntary passage of the survey and other factors (freedom of choice of voting place, the free exchange of keys between respondents belonging to one group of the survey). Otherwise, we risk getting feedbacks that do not reflect the real situation.
- Motivation factor. The voluntary participation in the survey is the most important factor for getting real results. One of the solutions is that, during the distribution of

keys the total number of people who want to take part in the survey is counted, and this number of keys is given to group, the members of the group can freely exchange keys. The remaining keys are deleted to prevent fraud. This mechanism allows to determine the relevance of the feedback service. Moreover, it provides more relevant indicators of the results of surveys without "unmotivated" respondents. It can also serve as a source of data for analysis of other possible problems (such as a low level of demand for the keys for voting shows distrust to the mechanism of anonymity of the service and therefore it says about insufficient explanation of the principles of service, or about the lack of understanding of the mechanism of the functioning of network and services in the network).

3 Possible Usage of KSU Feedback

3.1 Universal Service for Surveys / Voting

Firstly, let's consider the KSU Feedback as a hypothetical universal service for conducting surveys to find out all common strong and weak points of the service in order to deal with a variety of tasks. The authors understand that creation of universal service which deals with a diverse range of tasks is too difficult (as it is quite tedious procedure), and often impossible, but this part of the article is needed to understand what service is KSU Feedback, and for what purposes its use is inappropriate.

Thus, a hypothetical universal service for the surveys should have the following generic characteristics:

- It must have a sufficient number of types of issues to deal with universal problems.
- It must have ability to conduct normal / anonymous voting.
- It must have an intuitive interface, sufficient documentation for developers / users.
- It must have an API for working with all the basic data (surveys, questionnaires, questions, etc.) and perform all the basic operations (conducting a survey, obtaining of the results of the survey, creation of surveys / questionnaire / questions).
- The presence of widgets for integration on other sites (survey, view of the results, etc.).
- Ability to analyze surveys by the creators of the polls.
- It must support the export / import data (questions, polls, surveys) and use commonly used data formats.
- It must have the system of automatic notification of the participants of surveys / surveys creators / administrators of the various actions (creating of the survey, the expiration of the survey, the achievement of certain results, for example, 200 people voted and a certain score fell below 5, etc.) by e-mail, messaging within service, SMS, and so on.

Factors that prevent service KSU Feedback from becoming the universal service of surveys:

- Difficulty to identify the user (if the creator of the survey is intended to carry out normal polling).
- The current capabilities of API is not enough.
- It is not enough number of types of questions.
- There is no system (module) of import / export of issues / surveys / members, etc., analysts and analysis.

Also, we should note that during creation the service was not positioned as a universal service for the surveys and wasn't intended to have the means and tools to solve common problems, so we won't consider the KSU Feedback as a universal tool. KSU Feedback fills a target niche: obtaining of anonymous feedback from a specific group of persons. Further we consider just this use of the service.

3.2 KSU Feedback as Service for Conducting Surveys in Specific Areas

Universities, colleges, technical schools. Students have a great role in maintaining quality and enhancing learning due to their engagement in processes of internal quality assurance. Students' evaluation may bring about many potential benefits including opportunities to gather students' feedback on a non-face-to-face basis without fear of sanctions, to help educators understand the concept of institutional educational quality, to encourage teachers to identify areas of improvement and raise standard of teaching quality, to provide areas of research in teaching, to maintain a less intrusive form of classroom appraisal and to provide a broader and more objective base of evaluation in comparison to one which is undertaken by the department chairperson/ director. (Ting, 1998). Therefore it's argued that students have great impact in shaping improvement [8].

As it was mentioned earlier, KSU Feedback was developed and works as a means of feedback 'teacher-student', the service takes into account the specifics of the area [1]. The issue of transparency and accessibility of service is opened. At the time of writing of the article a free registration form was available on the service (<http://feedback.ksu.ks.ua/registration/>).

The advantages of using of the service (accept those already mentioned advantages in the form of teachers receiving feedback) as follows:

- The use of ready-made solution. It's unnecessary to waste time and resources to develop similar service.
- Anonymity for the student is very important factor for them as for respondents of survey.

Currently service is free and can be used to obtain an objective opinion of students about the quality of teaching, and it also can be used as tool for get feedback on other aspects of the institution (work, study, meals, lodging, etc.), so we recommend the management of educational institutions to pay attention on the service for possible future use.

Government institutions. We have identified government institutions as a separate item, as use of KSU Feedback (as well as use of any other private tools) in public institutions is rather difficult because of several reasons:

- inertness of thinking - as misunderstanding of the principles of functioning of services in particular, and the network as a whole, and the reluctance to change anything in an established system;
- lack of interest in obtaining objective feedback and results;
- legislative base - permission to use the service.

Despite all these difficulties while using the KSU Feedback you can get an objective assessment of the work of state employees by people who use their services [9], [10]. One of the main advantages of KSU Feedback for use in the state institutions is anonymity. Possible scenarios of the use of KSU Feedback in the state institutions have much in common with the scenarios used in commercial organizations (only the actors are changed), but specific details of the service in state institutions are not considered in this article because of low awareness of the authors of this specificity (levels of security clearance, the licensing of software to use , etc.).

KSU Feedback has great potential for use in the state institutions, but it is required further investigation of each area of use, taking into account the specifics of the organizations.

Commercial organizations. In comparison with state institutions commercial organizations have more prospects to use KSU Feedback, such as management of commercial companies are interested in obtaining reliable information about the quality of customer service (work with clients, loyalty to the old and new customers are main components of business sectors) [8], [10], [11], [12].

A common scenario of the service may look like this:

1. User1 receives services from User 2.
2. User2 receives a short link for voting online (getting links can be a volunteer - according to the customer, and can be a service by default - printed reference is given together with the main papers, or it can be printed on the back of a receipt, if the service provides a receipt). Anonymous key for voting is built into initial address (example: the source link - <http://feedback.ksu.ks.ua/voting/go/Ad34FdE44>, short link - <http://tinyurl.com/75eyjlv>). Transformation of the initial address into the shortest one serves for the convenience of reference input from the keyboard.
3. After finishing the results of the survey can be used for different purposes.

There are many variations of this scenario, such as evaluation of the service provided by website or by other service, etc. (provision of services by using Internet). In this case, the user can estimate the comfort of provided service (for different services you can generate different types of surveys with different questions such as: convenience, clarity, speed of service, the general impression, etc.), using the generated link to the survey after getting the service, or using the built-survey widget on the side of the client's site (in this case a client means the service provider (customer of service KSU Feedback is owner of the site-resource)).

3.3 Scenarios of use the KSU Feedback in a commercial company.

Scenario 1.

Respondents: customers.

What is evaluated: the level of service after receiving any service.

Objective: To obtain customer satisfaction, an analysis of problems in service (hardware, software, training, etc.).

Scenario 2.

Respondents: the staff of the organization.

What is evaluated: the level of leadership, working conditions, the overall satisfaction of the conditions, etc.

Objective: To assess the level of management (middle management), identification of problems, in tandem with the previous version of usage (survey of customers about staff), it can be a source of data for analysis and argument for the adoption of various personnel decisions.

Key features: anonymity of the employee.

Also we should note that the use of the service for a particular area may have some specific requirements. For example, during creation of a survey it is possible to deny access to the survey results for a certain period of time, if the operator of the service, to which the client can leave a response, has the right to view the survey results (for example to increase the level of customer service). This option is required to prevent deanonymizatsii client. Another possible requirement (eg for financial institutions), is the work of service over a secure protocol HTTPS. There are many possible modifications of the service to cover the specifics of a subject area and the main conclusion is the next - the service must be able to customize (settings) to the needs of a particular company. This can be achieved by:

- API service - for serious modifications, or integration with third-party applications the resources for use are necessary.
- The settings of various parameters on the service side - using the interface KSU Feedback configuration of parameters should be flexible.

That is why the service should be able to extend (to have the appropriate architecture), should have the possibility for flexible setting and API for working with master data. In the case when large organizations want to use KSU Feedback, they must have a module or system of analysts and analysis and / or the ability to export data using standard formats (without using the API). In addition, the following improvements are evident for comfortable work with the service using the above script:

- Ability to autogeneration of keys on request. In case when the number of survey participants is unknown (eg, when we do not know the number of served customers for the next week, but we can roughly estimate this).

So, let's form the requirements for working with commercial organizations (let's divide them into local and global according with approximate estimation of time for development / modification of functionality):

Global changes:

- A complete API.
- The ability of flexible customization using interface from the website service (show/hide results of vote after voting for respondents, show the results for user after some period of time (2 hours, 1 day, etc.), voting in the definition of polling days).
- Module (system) of analysts.
- Ability to export data.

Local changes:

- The ability to generate polls with unlimited number of keys or with the ability to add participants to the survey.
- Built-in widget for surveys.
- Types of questions and comments to the question (general survey) by the user's request.
- Ability to identify respondent at his request (the ability to leave contacts for the respondent for further communication and work with the respondent).

4 Conclusions

KSU Feedback is a service designed to deal with specific problems, and we should not consider it as a universal service for conducting surveys. However, the service solves task of anonymous voting among certain range of respondents. As a universal service of surveys, KSU Feedback requires serious modifications, but nevertheless it is suitable for obtaining feedback from the respondents keeping anonymity, and has great potential for use in commercial and government organizations, the basic scenarios of which were considered above.

The usage of the service at educational institutions allows teachers and the leadership of institutions to receive anonymous feedback from interested groups of respondents. Using of the KSU Feedback may pursue such goals as:

- Teachers obtain feedback on the quality of education (which encourages teachers to improve the quality of teaching).
- To obtain ratings of teaching of specific subjects.
- It allows you to create a sense of mutual accountability, student and teacher.
- To obtain feedback on the work of an educational institution as a whole (education, work, residence).
- The feedback analysis rapidly shows where a person needs to improve skills or has to acquire new knowledge.

References

1. Spivakovskiy, O.V., Berezovsky, D.O., Titynok, S.O. : ARCHITECTURE AND FUNCTIONALITY OF THE PROGRAMM COMPLEX "KSU FEEDBACK". In: 5th Issue Information Technology in Education, pp. 40--53. Kherson (2010) (in Russian)
2. Dybina, I.: Mathematical and statistical methods in empirical socio-economic research. Infa-M Finance and Statistics (2010) (in Russian)
3. Matcovsky, S., Marets, A.: Theory of Statistics. Second edition, stereotyped. K.: Knowledge. (2009) (in Russian)
4. Spivakovsky, O.V.: On the impact of information technology in education technology. In: 4th Issue Computer - oriented system of education, pp. 3--11. National Pedagogical Dragomanov University, red. Jaldak, M. (2001) (in Ukrainian)
5. Dichkivska, I.: Innovative educational technology. Akademydav (2004) (in Ukrainian)
6. Ward, P.: 360-Degree Feedback. Hippo Publishing Ltd (2006)

7. Customer Feedback - Why is it So Important?, <http://ezinearticles.com/?Customer-Feedback---Why-is-it-So-Important?&id=1327205>
8. The Importance of Customer Satisfaction in Relation to Customer Loyalty and Retention, <http://www.ucti.edu.my/wps/issue1/wp-06-06-paper.pdf>
9. Maurer, R.: Feedback Toolkit: 16 Tools for Better Communication in the Workplace, Second Edition. Productivity Press (2011)
10. Seashore, C.N., Seashore, E.W., Weinberg, G. M.: What Did You Say?: The Art of Giving and Receiving Feedback. Bingham House Books (1997)
11. Weitzel S.R.: Feedback That Works How to Build and Deliver Your Message. CCL Press (2000)
12. The Importance of Customer Satisfaction and Customer Engagement in Business Outcomes, <http://blog.peoplemetrics.com/the-importance-of-customer-satisfaction-and-customer-engagement-in-business-outcomes/>

Issues of Model-Based Distributed Data Processing: Higher Education Resources Evaluation Case Study

Olga Cherednichenko¹, Olga Yangolenko¹, and Iryna Liutenko¹

¹National Technical University “Kharkiv Polytechnic Institute”, Frunze st. 21,
61002 Kharkiv, Ukraine
{marxx75, olga_ya26, chlivi_68}@mail.ru

Abstract. Higher education resources are considered in this paper as a complex heterogeneous hierarchical system. The formal mathematical models for resources evaluation are suggested. The analysis of information and communication technologies applied in higher education establishments is conducted. The necessity of distributed hardware and software infrastructure for data storage and processing in the evaluation activities is shown and substantiated. The distributed data storage and processing architecture is presented.

Keywords. Distributed data processing, evaluation, information system, quality, higher education resources.

Key Terms. ProcessPattern, FormalMethod, Model, SoftwareComponent.

1 Introduction

Nowadays education quality is a crucial factor of both competitiveness of higher education establishments (HEE) and success of HEE graduates in their careers. That's why assessment of education quality is an urgent problem for management activities. The adequate education quality estimates can be the basis for correct management decisions and for realization of improvements. Moreover the estimates of higher education quality may be important for employers and for potential applicants as the main stakeholders of the system of higher education.

Education quality assessment is directly connected with education quality model. Various quality models are used in HEEs around the world [1]. Some of them are oriented on possibilities and results of business processes of HEE. For example, EFQM Excellence Model adopted for higher education introduces two groups of criteria: enablers and results [2]. Enabling criteria cover what the organization does, and the results criteria cover what the organization achieves.

As a rule, HEE plans and manages internal resources in order to support its policy and strategy, and the effective operation of its processes. Resources are the means that

provide HEE’s functioning. During planning and managing of resources HEE balances its current and future needs.

There are basic and supporting processes in HEE. Educational process is the process which realizes the main function of HEE. It supports university’s activities intended for delivering knowledge and skills to students. The resources of educational process include the staff, material and technical facilities and courseware. For example, to teach students some subject, first of all, it is necessary to appoint competent lecturer and a tutor. Secondly, some well-equipped room for lessons should be found. Handbooks, guidelines and other courseware are necessary for representation of teaching material. And finally, to support the educational process such information resources as curricula, announcements and other useful information should be available in a convenient way, for example via the local network or the Internet.

The educational process is justified by academic curricula for different qualifications, such as bachelor and master. The basic unit of academic curriculum is a discipline (fig. 1). Every discipline is described by its syllabus. Syllabus determines all necessary resources for discipline teaching. All HEE’s resources are distributed among different units and are allocated on vast territories (fig. 2). Moreover, some resources are in common usage of HEE’s units.

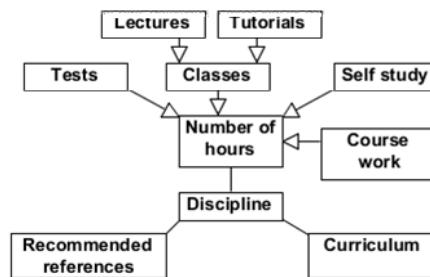


Fig. 1. Discipline structure.

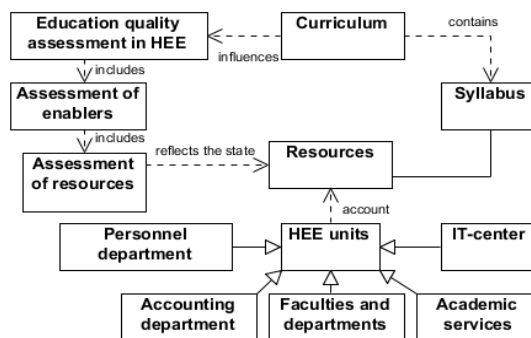


Fig. 2. Resources assessment

So teaching disciplines requires large amount of various resources that are shared, commonly used and accounted by different departments of HEE. Under these conditions assessment of resources that support educational process seems to be a complex problem. The solution of this problem implies collecting data from different sources, its storing, processing and taking decisions about resources updating.

The rest of this paper is organized in the following way. Section 2 classifies information systems in HEE and characterizes mathematical methods used for their realization. Three mathematical patterns that support resources quality assessment are considered in Section 3. Section 4 represents discussion about distributed and centralized data storage and processing. Grid-based architecture is suggested. Section 5 presents conclusions and prospect on future work.

2 Analysis of Information and Communication Technologies in Higher Education

The HEE's functioning is based on various processes, therefore HEE has different information systems (IS) that support these processes. For example, there can be distinguished information systems of administrative and financial management, educational process management and support, scientific research management, information resources management [3]. HEEs use either commercial IS or those which are elaborated by universities for their particular needs [3]. Often commercial solutions do not take into account all peculiarities of definite HEE's functioning, so they can't satisfy all requirements. To elaborate its own software HEE usually has a lack of resources and highly qualified staff. However both types of IS are used and the problem is to integrate them and to provide efficient interaction between solutions of different providers.

The work of all university's IS is based on mathematical models used for solving management problems. Problems of data accounting are managed by means of data bases (DB). Queries to DBs are formed with the help of relational algebra [4]. Data stored in DBs is processed basing on statistical methods which include correlation, variance, discriminant, factor, cluster and other kinds of analyses [5]. Since testing is an important and specific educational problem, processing of testing results is supported by Classical Test Theory [6] and Item Response Theory [7]. HEEs rating and various assessment problems are solved with the help of expert methods [8].

As we can see there are different tasks that should be automated in the university. Different mathematical models and IS are used to solve these tasks. All existing and elaborating IS are intended to the common purpose – to improve university's functioning. From the management point of view all IS should exist in the common informational space with distributed data storage and processing. So we can make the conclusion that it is necessary to develop distributed hardware and software infrastructure composed of heterogeneous resources owned and shared by multiple administrative units which are coordinated to provide transparent, dependable and consistent computing support to a wide range of applications.

Since quality can be considered as one of the main goals of HEE's management, the resources evaluation subsystem has to be a component of HEE's software

infrastructure. Resources quality influences higher education quality directly. Our task is to extend the functionality of existing IS by implementing resources evaluation.

Quality monitoring and evaluation (M&E) is a part of the management process. There are two basic types of monitoring: implementation-focused and results-based [9]. Implementation-focused M&E is oriented on inputs, activities and outputs of system's functioning. Results-based M&E focuses on goals and results giving the evidences and explanations of existing tendencies. Our research is oriented on results-based M&E. In the given work we consider the evaluation process which provides management with necessary estimates towards defined management outcomes.

3 Model-Based Resources Evaluation

We consider the system of university's resources as a complex heterogeneous hierarchical structure, which has a large number of parameters. HEE resources are used, controlled and evaluated by different departments and units. We found distributed structure of data sources providing partial estimates of resource units. We distinguish three basic tasks connected to the university's resources M&E. They are resources comprehensive quality assessment, internal licensing audit and resources usage performance evaluation. These tasks are interconnected and related to quality assessment from different points of view.

Higher education licensing, i.e. the process of granting permissions to provide certain educational services, is an important process of public administration. The licensing process is carried out on a regular basis (the license validity is limited), it requires processing of large data volumes, and supposes that the information related to HEE is available for public access. This information in particular may include curricula, syllabus, university infrastructure, etc. The internal audit is required to prove the sufficiency of existing resources and their quality to obtain license. During the internal licensing audit HEE is estimated according to the license conditions [10].

Resources usage performance can be evaluated through the set of indicators – different for each type of resources. For example, performance of capital resources is measured via loading factors, profitability ratio, usage intensity coefficients. The efficiency of human resources usage is determined by labor performance, annual number of workers, economy or excess expenditure of salaries.

The main goal of management in university is improvement of education quality. Resources of the university provide the possibility to reach this goal. Updating university resources is the most realistic way for improvement of education quality. Therefore resources comprehensive quality assessment is the basis of continuous quality improvement. It allows finding the resources that have to be updated, to evaluate their influence on educational process, and to elaborate resources updating projects.

Partial and aggregate indices are used for decision making in all mentioned tasks. Therefore M&E IS must have the model of transition from a set of partial to aggregate estimates. To solve this problem we suggest applying Resources Network System (RNS) [11].

The structure of evaluation system can be presented as an oriented acyclic graph $G_k = (K, A_k)$, where K is a set of nodes, $A_k = (a_{ij})$ determines the directions of arcs, that connect the nodes. The given graph has two types of nodes: node-entries $K^0 = \{k_j \in K : \forall k_i \in K, a_{ij} = 0\}$ and node-aggregates $\tilde{K} = \{k_j \in K : \exists k_i \in K, a_{ij} = 1\}$. Node-entries reflect the partial estimates of resources. The aggregation logic is defined by particular task. Each aggregate is associated with some composite function, or estimator.

In this paper we pay our attention to the task of resources comprehensive quality assessment. We consider direct and reverse tasks of comprehensive assessment. The direct task is the computation of comprehensive assessment value on the assumption of known values of estimators. The reverse task is the determination of values for node-entries on the assumption of user-defined comprehensive assessment.

The ratio scale is used for both types of estimates. We suggest using Qualimetry Theory (QT) for partial estimates computing. The QT provides generalized principles of quantitative assessment of quality of objects of any nature [12]. That's why we choose the method of qualimetry to develop node-entries assessment methodology. To aggregate partial estimates we use the following convolutions as estimators: weighted average arithmetical, weighted average geometric, weighted guaranteed result, and weighted dominating result index. We use different types of estimators because of heterogeneity of resources, different influence of various resources types on the comprehensive resources quality. These convolutions require experts' judgments.

We use the QT-based approach for partial estimates calculation which includes the following steps.

1. Situation assessment is defined, i. e. description of the conditions and goals of the assessment, application of those estimates are identified.
2. The properties tree is constructed. It reflects the hierarchically ordered set of features of the object and allows to fully describe its quality. The procedure of properties tree construction is based on a number of claims detailed in the QT [12].
3. For each simple and some complex features it is necessary to assign appropriate indices. Every index is associated with measurement scale, reference and rejection values. The absolute values of all indices must be converted into relative ones.
4. Weighted coefficients are calculated. We suggest defining weighted coefficients using pairwise comparison. The weighted coefficients calculation is based on the method of eigenvector [13].
5. Partial quality assessment is calculated based on one of the weighted average methods. We use weighted coefficients and relative values of indices obtained on the previous step.

Thus, all resources, that support educational process in HEE, are associated with an oriented graph. Every node of this graph reflects individual or group estimates of resources. We propose pattern-based data processing and distinguish the following patterns.

1. The QT-based Partial Assessment Technique (PAT) is a pattern for assessment of separate resources units. Based on this pattern the evaluation procedure is formed, which results in partial estimate.

2. The Comprehensive Assessment Technique (CAT) defines the grouping of estimates of resources units from different points of view. We suggest to represent CAT as a graph which node-aggregates are associated with one of the convolutions mentioned above.
3. Weighted coefficients Calculation Technique (WCT) provides expert judgment method based on pairwise comparisons. This pattern includes the procedure of processing pairwise comparisons and calculation of weighted coefficients vector.

Resources evaluation is considered on three stages. They are pre-processing, data processing and interpretation of results. These stages are repeated for each evaluation task. The pattern is chosen depending on the task that has to be solved. Initial data sources are determined by RNS. Pre-processing stage includes data extraction from initial sources, collection of required data and its transformation. Data processing stage is totally defined by the chosen pattern. On the results interpretation stage the obtained estimates must be explained from the point of view of the considered task.

Based on the above discussion we suggest creating the resources evaluation software.

4 Architecture of Resources Evaluation IS

Currently the two common ways of data storage are centralized and distributed approaches. Centralized data storage applies that a DB is located at a single server. The data access is executed with the help of remote query or transaction. This is the simplest approach for realization and maintenance. The disadvantage is that the DB would be unavailable for remote clients if connection errors occurred. Also database located on one server has a limited volume of memory. Since all queries are sent to a single server, there are some obvious constraints in time delays and costs of connection support.

Distributed approach supposes that data is stored on multiple servers. This allows increasing of DB volume. Since many queries are satisfied by local DBs, the response time and the costs of query processing are decreased while availability and reliability are increased. The disadvantage of distributed data storage is that some queries and transactions may need the access to all servers which increases the response time and the costs. Another issue is that all clients have to have information about data location in distributed DB. This approach is compatible with common usage of local and global networks.

Distributed data storage and processing provides efficient work with rapidly changing information that is used by various clients. It supports a large number of cooperating clients which collect, register, store and deliver information. Distributed data storage prevents servers from overloading by distributing data among different computers. Also it provides an access to a huge volume of information stored in the system.

One of the most convenient approaches of distributed data storage and processing is grid technology. There are three types of grid systems: computational, data and network [14]. A computational grid has the processing power as the main computing resource shared among its nodes. A data grid has the data storage capacity as its main

shared resource. Such grid can be regarded as a massive data storage system built up from portions of a large number of storage devices. Network grid has as its main purpose to provide fault-tolerant and high-performance communication services.

In this work we suggest using data grid architecture for M&E software in HEE (fig. 3). Data stored by various departments and units is shared through the Internet or local network. Data processing is conducted by application servers available for different users via their desktop applications. M&E software services are located on application servers. Special management system is required to control, coordinate and synchronize distributed data storage and processing.

The PAT, CAT and WCT services are based on the corresponding patterns described above. The data collection service provides tools for data extraction from distributed database. These services can be used by quality evaluation software. If we need to evaluate resources quality, this software must store RNS model and data sources for partial estimates. Such software must support pre-processing and interpretation stages. On the data processing stage it calls the necessary services.

We suppose that Open Grid Services Architecture is the suitable solution for distributed hardware and software infrastructure. This will be a base for university's IS integration.

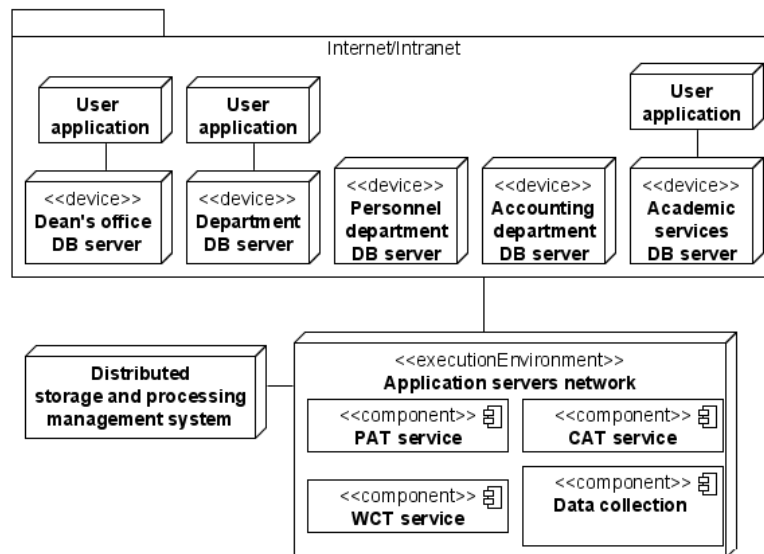


Fig. 3. Distributed data storage and processing architecture.

5 Conclusions and Future Work

We considered a case study of comprehensive resources quality assessment. Higher education resources have distributed and heterogeneous structure. This leads to the

idea of distributed data storage and processing. The given paper generalizes methods of quality assessment into several patterns. The suggested approach allows quality assessment of heterogeneous objects and has comprehensive facilities for construction of quality measurement tool. The architecture of information system of resources evaluation is based on distributed data grid architecture. Such approach may be generalized for evaluation of different objects, not only resources and not only in higher education establishment.

Our future work includes elaboration of formal models for solving two remaining tasks – internal licensing audit and resources performance evaluation. The further researches will be devoted to detailed architecture design and software construction.

References

1. Existing models of education establishments' quality systems, <http://ru.ict4um.edu.ru/lib/euro/model> (in Russian)
2. EFQM Excellence Model. Higher Education Version 2003. Centre for Integral Excellence, Sheffield Hallam University (2003)
3. Krukov, V.V., Shahgeldian, K.I.: Corporate Information Environment of University: Methodology, Models, Solutions. Dalnauka, Vladivostok (2007) (in Russian)
4. Date, C.J.: An Introduction to Database Systems. Addison Wesley (2003)
5. Ross, S.M.: Introduction to Probability and Statistics for Engineers and Scientists. Elsevier Academic Press, London (2004)
6. Steyer, R.: Classical (Psychometric) Test Theory, <http://www.metheval.uni-jena.de/materialien/publikationen/ctt.pdf>
7. Reeve, B. An Introduction to Modern Measurement Theory, <http://www.appliedresearch.cancer.gov/areas/cognitive/immmt.pdf>
8. Meyer, M. A., Booker, J. M.: Eliciting and Analyzing Expert Judgment: A Practical Guide. SIAM, Philadelphia (2001)
9. Kusek, J.Z., Rist, R.C.: Ten Steps to a Results-Based Monitoring and Evaluation System: a Handbook for Development Practitioners. The World Bank, Washington, DC (2004)
10. Cherednichenko, O., Kuklenko, D., Zlatkin, S.: Towards Information Management System for Licensing in Higher Education: An Ontology-Based Approach. In: Mayr, H.C., Karagiannis, D. (eds.) Information Systems Technology and its Applications 6th International Conference ISTA 2007. LNI, 84, pp. 33--42. GI, Bonn (2007)
11. Cherednichenko, O., Timchenko, K., Liutenko, I.: Technology of Quality Comprehensive Assessment (for Resources in the University by Example). In: Vestnik of Kherson National Technical university, vol. 2 (41), pp. 451--455. KNTU, Kherson (2011) (in Russian)
12. Azgaldov, G.G.: Theory and Practice of Goods Quality Assessment (Basics of Qualimetry). Economics, Moscow (1982) (in Russian)
13. Saaty, T., Vargas, L.: Decision Making with the Analytic Network Process. Economical, Political, Social and Technological Applications with Benefits, Opportunities, Costs and Risks. Springer (2006)
14. Hose, K., Schenkel, R.: Distributed Data Systems: Introduction, http://www.mpi-inf.mpg.de/departments/d5/teaching/ws10_11/dds/slides/DDS-1-print.pdf

Tested Approach for Variability Management Enhancing in Software Product Line

Andrii Kolesnyk¹ and Olga Slabospitskaya¹

¹Institute of Software Systems of NAS, Akedemika Glushkova st., 40, Kiev, Ukraine
{kolesnyk, slabospitskaya.olga}@gmail.com

Abstract. The paper declares a novel approach for the process enhancing of managing Variability – the ability of a software system or artefact in Software Product Line (PL) to be extended, changed, customized or configured for use in a specific context – with the proper quality characteristics to mitigate its current limitations. New Variability Model and Management Functions to process its element are proposed as this process Core. The model consistently represents variabilities both in PL structure and artefacts across all PL development stages and stakeholders' viewpoints along with the dedicated assessment submodel. The functions compose separate actions as to variability into the single cycle like Doeming Plan-Do-Check-Act one where decisions should be rational. Presented successful Case Study purposes at the Core proposed testing along with the dedicated Configurator implemented in instrumental and technological complex just developed in the Institute of Software Systems of NAS.

Keywords: Software Product Line, Variability Model, Variability Management, Reusable Asset, Configuration.

Key Terms: Model-based software system development: Method, Model, Methodology, Process, Software System.

1 Introduction

Effective and efficient large, complex and multi-purposed software systems composition from more simple reusable assets was one of the challenges being addressed in the research project named “Theoretical Fundament of Generative Programming and Means of its Support”(2007-2011, № 0107U002205) [1, 2] just accomplished in the Institute of Software Systems of NAS of Ukraine. Over the project Software Product Line (PL) Engineering [3] has proven to be the promising paradigm to produce a diversity of high-quality similar-but-different products with limited time and efforts.

But it was at once well-recognized that the key success factor in PL Engineering is a proper management of both the two kinds of variability disambiguated by Metzger et al. [4]. The first, specific PL variability, describes properties and qualities that should vary between the systems of the PL and that should not. In return, the second

one is single Software variability – i.e., the ability of a software system or artefact to be extended, changed, customized or configured for use in a specific context.

However, just now researchers' efforts concentrate foremost on variability modeling [4, 5] and implementing [3], while challengeable problems of its planning and evolving less attention are paid. One of perspective frameworks to consistently cope with them is COVAMOF [6] determining whether, when and how software variability in PL should evolve with special meta-model and method for its assessment.

The paper pursues the same goal but for both the above kinds of variability. It proposes dedicated Variability model and Management Functions based on its estimates as the Core of an empirical approach for PL Variability Management process defining that is enhanced with appropriate quality characteristics. The Core is tested with the dedicated Configurator elaborated within the research project above.

2 Variability Issues in PL

Variability items to be managed. Let fix, to use hereafter, the definitions of basic Variability items that have to be manage over PL development and therefore need to be explicitly modeled, following up the origins [3, 4, 6].

The first such item is a variation point. It is an abstraction that identifies location in software system or artifact at which a choice can be made between values or variants. As Deelstra et al. [6] note, it is not by-product of design and implementation of variability, but answers the question, what does vary, being therefore identified as central element in managing variability. Each variation point is associated with a value, zero or more variants. Variation points are categorized to five basic types such as: optional (zero or one variant out of $1, \dots, m$ associated variants), alternative (1 out of $1, \dots, m$), optional variant ($0, \dots, n$ out of $1, \dots, m$), variant ($1, \dots, n$ out of $1, \dots, m$), and value (a value that can be chosen within a predefined range).

A variant is thus the second Variability item answering the question, how does vary the variation point it is associated with [4].

The third one is dependency [3, 6]. It specifies a function of how the choices at the variation points in the PL influence a system property value, e.g. quality attribute, as well as the valid range for this value. The last one, namely constraint, is a predicate that defines possible interrelations between various variation points and variants.

Variability Levels in PL development. Thorough study of PL Development process [2, 3] experiences five possible types (t) of variation points and variants corresponding their Lyfe Cycle over PL Development:

- Features, i.e. abstract concepts reflecting commonalities and variabilities of software products in PL relevant for some Stakeholder that might represent a technical function, a function group or a non-functional characteristics ($t = 1$)
- Requirements as to Software Products ($t = 2$)
- Architecture components ($t = 3$)
- Database tables ($t = 4$)
- Software artifacts ($t = 5$).

Target characteristics for Variability Management enhancing. Based on the experience and ideas formed during the above Institute of Software Systems of NAS research project (№ 0107U002205) [1, 2], four Berg et al. [5] essential quality characteristics are chosen to adopt as a target for the Variability Management process to be defined. These are consistency, scalability, traceability and visibility.

Consistency means that Variability should be handled the same way at all above levels of abstraction and across all PL development phases to reduce the ambiguities that might occur when using different methods for managing variability at different abstraction levels. Scalability prescribes that the methods used should be easily applicable for both the single component and a large complex system. In turn, traceability requires that Variability items at different levels of abstraction and across development phases should be explicitly linked both upwards and downwards to simplify PL evolution and maintenance. Lastly, visibility presupposes understandable representations of all Variability items in appropriate and intellectual user interface.

3 An Approach Proposed to enhance Variability Management

An approach proposed is simply to define a Core of Enhanced (i.e. possessing the above target characteristics) Variability Management process, then continuously test it under various stressing conditions and refine accordingly to the lessons learnt.

The Core is formally a tetrad of a sets open for expanding

$$C = \langle AS; FN; ENV; DM \rangle; ENV = \langle VM; RP; VP; CP \rangle, \quad (1)$$

where *AS* denotes a priori assumptions as to PL development organizing; *FN* is the set of Generic Functions for PL Variability Management Process; *ENV* is the environment for *FN* operation including dedicated Variability Model (*VM*) described hereafter, PL core assets Repository (*RP*) and profiles both for PL variability with *VM* (*VP*) and for core assets reuse over PL development (*CP*); *DM* is the set of Demands the Functions should meet to enable the target quality characteristics be really attained.

Initially *AS* in (1) assumes PL development to be the series of unified production rounds interchanging with the rounds of PL environment actualization.

In the *FN* set, four target Variability Management functions are elicited as generic through comparative study of popular Variability Management process templates [2-4, 6] within the perspective of Doeming's PDCA Management Cycle [7]. These are informed and consistent Variability Planning, Implementing in PL artifacts, all-aspect Controlling and Evolving up to both retrospective and current elements of *VP* and *CP*. They serve due rationales for appropriate managerial decisions over *FN* processing. All necessary technological prerequisites as well as initial *VM* and *RP* are created with the fifth function of Variability Management Initiation.

The *DM* set from (1) composes consistency, scalability, traceability and visibility demands being inspired the title target characteristics and also the additional demand of rationality for all decisions being made under the functions *FN* processing. To

meet this demand is the main purpose of *VM* and both the above variability kinds assessment method with it that enables *VP* and, particularly, *CP*.

4 Consistent and Traceable Variability Model

Let's particularize the above target characteristics and demands *DM* from (1) to fix inspired demands the dedicated *VM* should meet to pursue its purpose in (1):

- uniform, consistent and traceable representation for all the variety of variability items and their interrelations over all the stages both for PL Domain and Application Engineering processes [2-6, 8] as well as for all functional segments from PL scope and also for all its stakeholders' viewpoints
- traceable notations usage for PL artefacts modelling appropriate to their types
- explicit identification of commonalities and variabilities across all PL development artefacts, stages and stakeholders' viewpoints
- sound, informed and consistent PL variability profile assessment.

Relevant Variability Model is defined [8] to be a hybrid structured triple

$$VM = \langle SV; AV; EV \rangle, \quad (2)$$

where: submodels *SV* and *AV* represent variability in PL structure and its artifact; *EV* is an integrated submodel for informed and consistent variability assessment.

The first submodel *SV* in (2) gives the formal representations of all the features from PL scope, both commonalities and variabilities, artifacts to implement them and their links on the base of feature modeling approaches [2-4, 6]. It is a structured tuple

$$SV = \langle G_1; \langle \langle G_t, TR_t \rangle, t = 2, 3, 4, 5 \rangle; CN; DP \rangle, \quad (3)$$

where: $G_t = \langle F_t, LF_t \rangle$ is the graph where unique identifiers of *t*-typed PL artefacts (i. e. features, requirements, architecture modules, database tables, software components and tests) are nodes sets (F_t) linked through obligatory and variant binding (LF_t); TR_t is bilateral traceability links between the nodes of G_{t-1} and G_t graphs; *CN* and *DP* are the predicates on $\otimes_{t=1, \dots, 5} F_t$ representing PL constraints and dependencies.

In turn, the second submodel *AV* in (2) provides unified formal representations for all PL software products currently located in repository, being developed or might be developed eventually within current PL scope together with their development products (requirements etc.). To explicit reflecting the elements of *SV* (2) model in PL software products, any *t*-typed artefact is formally seen as cross-cutting "fragment" of *SV*. It is bounded with continuous upwards – downwards traceability links TR_t from (3), which interrelates this artifact with the features it should implement and the final software product. The model *AV* is a structured tuple

$$AV(id_t) = \langle g_1; \langle \langle g_u, tr_u \rangle, u = 2, \dots, t \rangle; \langle \langle p_u, tr_u \rangle, u = t + 1, \dots, 5 \rangle; cn; dp; s \rangle, \quad (4)$$

where: id_t is the modeled artefact's unique identifier; g_u and p_u are subgraphs of G_u from (3) representing the artefacts that are implemented with id_t and, respectively, implement it over PL development; $tr_u \in TR_u$ are subsets of traceability links between the nodes of g_{u-1} and g_u ; cn and dp are the limitations of CN and DP on Cartesian product of g_u ; and p_u nodes sets and s is the artefact's current state (e.g. "core asset" etc.).

Note that each of the five "horizontal" planes, implicitly defined with formulas (3), (4), reflects the viewpoints both at PL and artefact variability by specific PL Stakeholders group being represented over PL development with the proper-typed artifacts – from the customers' features at the first level (G_1, g_1) downwards to the programmers' and testers software and tests at the fifth one (G_5, g_5).

The third submodel of SV model expands Metzger's Orthogonal Variability Model [4] with a novel dimension of sound quantitative variability estimates $vp \in VP$ from (1). They quantify the level of PL variability adequacy within the perspectives of both PL customers' business needs in its products' functions and PL developers' requirements as to them. For the estimates' plausibility and consistency to increase, universal preferences model such as Bayesian Net, Analytical Hierarchy and Value Tree should be configured up to the assessment situation [9]. It is also a triple

$$EV = \langle \langle VL, VR \rangle; VP; VAR \rangle; VP = \langle VpR, VpA, VpE, VpB \rangle; VAR = \langle VR, VA, VE, VB \rangle, \quad (5)$$

where: VL and VR are subsubmodels both for integrated variability adequacy level in a whole and, respectively, for its sublevels corresponding the artifacts' types; VpR, VpA, VpE, VpB are the sets of variation points in SV (3) model; VR, VA, VE, VB are the sets of variants for these variation points.

5 A Case Study to Test the Variability Management Core

Here the probe implementation of PL artefact variability model AV (4) is considered for simple domain of quadratic equations solving. While classical feature diagram [4, 6] clearly demonstrates the variety of variability items at the feature level, it's quite difficult to implement it in specific application in PL repository. Instead MS Visual Studio Windows Workflow Foundation (WWF) enables more information about AV with appropriate diagram (see Fig. 1).

Let's explain how such variability might be visualized and on-line managed with that WWF diagram. Note that the letter A at the Fig. 1 connected with the "plus" pictograms denotes the variation points that might be filled with the reusable assets as their variants while the letter B denotes possible variants. In the case at hand the "Discriminant" component contains simple code to find discriminant ($D = b * b - 4 * a * c$), implemented by the developer responsible for producing PL core assets. Depending on its operating outcome, there are three scenarios (use case):

$D > 0; D = 0; D < 0$ and three corresponding assets for them: “TwoRoots”, “ExactlyOneRoot”, “NoRoots”.

When using WWF, Visual Studio environment don't prohibit the developer from binding any reusable assets and variation points. In other words, we need a dedicated software tool that should support both the *VM* proposed (2)-(5) forming and actualizing and application configurations changing based on *VM* and reusable assets.

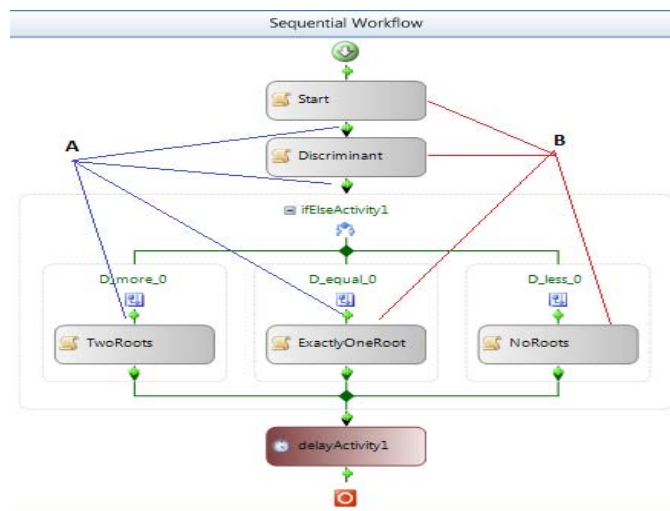


Fig. 1. Sample artefact variability model and reusable components are represented.

Trial prototype of such a tool, named Configurator, has been implemented within instrumental and technological complex just developed in the Institute of Software Systems of NAS [2, 10]. It purposes at configuring a diversity of similar-but-different applications from reusable software components and also at expanding and modifying applications with variation points and variants [1, 2, 8] based on WWF [2, 10, 11]. An interim result of configuration process with the Configurator is XML file shown beneath. It is an instruction for executable file compiling to create the target application.

```
<SequentialWorkflowActivity>
  <CodeActivity x:Name="Start"
  ExecuteCode="codeActivity1_ExecuteCode" />
  <CodeActivity x:Name="Discriminant"
  ExecuteCode="codeActivity1_ExecuteCode_1" />
  <IfElseActivity x:Name="ifElseActivity1">
    <IfElseBranchActivity x:Name="D_more_0">
      <IfElseBranchActivity.Condition>
        <CodeCondition Condition="WorkMeth1" />
      </IfElseBranchActivity.Condition>
      <CodeActivity x:Name="TwoRoots"
      ExecuteCode="codeActivity1_ExecuteCode_2" />
    </IfElseBranchActivity>
    <IfElseBranchActivity x:Name="D_equal_0">
```



```

<IfElseBranchActivity.Condition>
  <CodeCondition Condition="WorkMeth2" />
</IfElseBranchActivity.Condition>
<CodeActivity x:Name="ExactlyOneRoot"
ExecuteCode="codeActivity2_ExecuteCode" />
</IfElseBranchActivity>
<IfElseBranchActivity x:Name="D_less_0">
  <IfElseBranchActivity.Condition>
    <CodeCondition Condition="WorkMeth3" />
  </IfElseBranchActivity.Condition>
  <CodeActivity x:Name="NoRoots"
ExecuteCode="codeActivity3_ExecuteCode" />
</IfElseBranchActivity>
</IfElseActivity>
<DelayActivity TimeoutDuration="00:00:05"
x:Name="delayActivity1" />
</SequentialWorkflowActivity>

```

The configuration process producing this XML file is initiated through the special chart processing with WWF tool in Visual Studio environment (see Fig. 2).

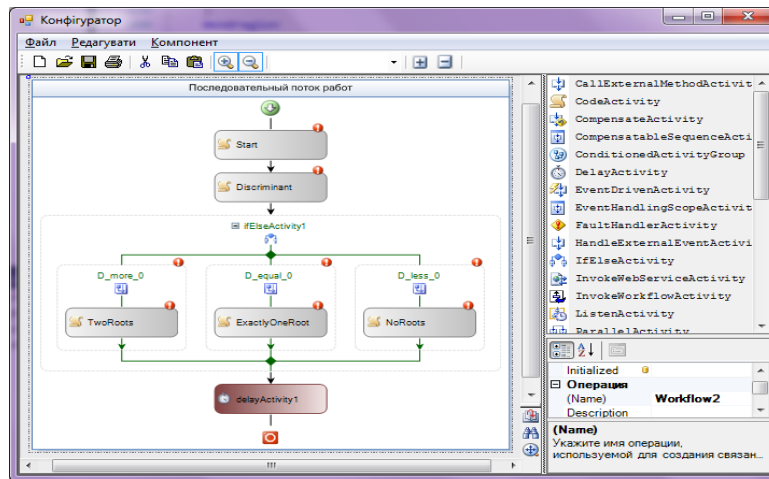


Fig. 2. Processing the reusable components by the Configurator is depicted.

Note that the application created is variable i.e. enables square equation solving under various conditions prescribed.

6 Conclusions

A novel Variability Model and generic Functions are elaborated for its enhanced (i.e. informed, consistent, scalable, traceable and capable to visualize the variability) support whole over PL development. The Model uniformly and consistently

represents all Variability items across all the relevant stakeholders' viewpoints and over all abstraction levels both in PL structure and artifacts. It also includes dedicated submodel for informed and consistent variability assessment. In turn, the Functions – Variability Planning, Implementing in PL artifacts, all-aspect Controlling and Evolving up to assessment results are serviced with Initiation one to initially create the above Model.

An approach is declared to construct Variability Management process being enhanced in the above sense. It prescribes to couple the Model and the Functions as a priori process Core, then continuously test and refine it up to the lessons learnt.

To attempt such a testing, trial Workflow-based Configurator is implemented within the instrumental and technological complex just developed in the Institute of Software Systems of NAS to effectively produce complex systems from the assets. Based on successful Case Study of sample product variant deriving, the authors now update their approach to support all the Functions and fulfil an industrial Case Study.

References

1. Lavrisheva, E.: Generative Programming of Software Products and Their Families [in Ukrainian]. In: Problems in Programming, vol. 1, pp. 3--16. Kiev (2009)
2. Lavrisheva, E., Koval, G., Babenko, L., Slabospitska, O., Ignatenko, P.: New Theoretical Foundations of Production Methods of Software Systems in Generative Programming Context [in Ukrainian]. Electronic monograph, in: UK-2011, vol. 67. Kiev (2011)
3. V. d. Linden, F., Schmid, K., Rommes, E.: Software product lines in action: the best industrial practice in product line engineering. Springer, Heidelberg (2007)
4. Metzger, A., Heymans, P., Pohl, K., Schobbens, P.-Y., Saval, G.: Disambiguating the Documentation of Variability in Software Product Lines: A Separation of Concerns, Formalization and Automated Analysis. In: 15th IEEE International Requirements Engineering Conference, pp. 243--253. IEEE Press, New York (2007)
5. Berg, K., Bishop, J., Muthig, D.: Tracing Software Product Line Variability – From Problem to Solution Space. In: Proc. of SAICSIT'2005, pp. 111--120. (2005)
6. Deelstra, S., Sinnema, M., Bosch, J.: Variability assessment in software product families. In: Information and Software Technology, vol. 51, pp. 195--218. Elsevier (2009)
7. Walton, M.: The Deming Management method. Dodd, Mead. New York (1986)
8. Lavrisheva, K., Slabospitskaya O., Kolesnik, A., Koval, G.: The Theoretical View for Software Family Variability Management [in Ukrainian]. In: Bulletin of University of Kiev. Series: Physics & Mathematics, vol. 1, pp. 151--158. Kiev (2011)
9. Lavrisheva, E., Slabospitska, O.: An Approach for Expert Assessment in Software Engineering. In: Cybernetics and Systems Analysis, vol. 45, no. 4, pp. 638--654. SpringerLink (2009)
10. Lavrisheva, E.: Instrumental and Technological Complex for Developing and Learning Aspects of Software System Development [in Ukrainian]. In: Bulletin of NAS of Ukraine, vol. 3, pp. 17--27. Kiev (2012)
11. Kolesnik, A.: Approaches to Configure Reusable Assets [in Ukrainian]. In: Problems in Programming, vol. 4, pp. 63--71. Kiev (2011)

**I.IV Methodological and Didactical Aspects
of Teaching ICT and Using ICT in
Education**

Motivating Students and Improving Quality of Learning Using Peer-Reviews

Vadim Ermolayev¹, Natalya Keberle¹, and Sergey Borue¹

¹ Zaporozhye National University, Zhukovskogo st. 66 69063 Zaporozhye Ukraine
vadim@ermolayev.com, nkeberle@gmail.com, bsu@znu.edu.ua

Abstract. This paper reports about a pedagogical experiment at Zaporozhye National University (ZNU) aiming at improving motivation and learning quality in Computer Science Bachelor programme. The major novelty in teaching and learning practice introduced in the experiment was the use of peer evaluation for the assessment of coursework reports in two disciplines – one in the II-nd and the other in the IV-th year of study. The results were compared to the historical data collected in the previous 3-4 years. Our experiment proved that exploiting students' aspirations for informal leadership and incurred competition constructively is effective and yields some increase in motivation to learn and learning quality. The assessments were also subjectively regarded as more clear and better justified by the students involved in the experiment. A good side effect is also that the students learn the working patterns of the professionals in their field broadly used in academia and industry for making qualitative and unbiased peer evaluations.

Keywords. Motivation, learning quality, peer evaluation, Computer Science, coursework.

Key Terms. Academia, TeachingProcess, Characteristic, QualityAssuranceProcess.

1 Introduction

Recent higher education experience reveals a substantial decrease of the popularity of University education and degrees reflected for instance in the decrease in degree completion rates [1]. Researchers analyzing the reasons for this decrease point out: (i) the rise of pragmatic attitudes to education in life planning among young people; (ii) the trend for devaluation of a University degree as a factor facilitating to employment and career development. As a result and because of the concurrent demographic and economic crises a substantial decrease of interest to quality learning among University students is observed. This observation is supported by the decrease in the student numbers and their grades. Consequently, the employers suffer from a decreased quality of the graduates.

Academia can not remove or relax demographic or economic factors unfortunately. Hence, the only feasible way of keeping academic performance at a competitive level is focusing on the stimuli for their students based on more social than purely pragmatic basics. For example, exploiting the value of informally assessed professional capability and leadership in student groups may be an effective way of stimulating spending more effort in learning.

The research presented in this paper aims at finding out such stimuli for Computer Science students based on their attitude to informal leadership grounded in professional competencies. The idea behind our pedagogical experiment was to place the subjects in an environment which is similar to professional and offer them to peer-evaluate their individual work. Hence, the higher the grades a person gets from his or her peers in such an evaluation – the higher becomes the professional reputation of the person in the group, making him or her informal leader in the group of the peers.

In fact the approach we have taken is not new and has been effectively exploited in the academic world as a peer evaluation mechanism as well as in social networks for forming communities of interest and building social reputation for the individuals in these communities. Such stimuli are qualified as **solidary** (in contrast to **material**) **incentives** [2] i.e. intangible rewards from the act of being a part of a group having coherent interests. In our research we build upon the mechanisms and tool support adopted from the mentioned domains. We involve students in peer evaluation of their individual coursework assignment reports similarly to that of reviewing conference papers. We measure their qualification by: comparing evaluations by peers and instructors – assignment results; and measuring deviations between their individual scorings and the mean values – reviewer competence. The anonymized results are then made available to the group.

We have observed that being an evaluator for the peers' work proved to be a noticeable incentive for the subjects who took part in our pedagogical experiment. Consequently the degree of active involvement and the quality of individual assignment results have increased substantially, in particular for the group in the last year of our Bachelor programme in Computer Science. This observation is backed up by the results presented in Section 4.

The rest of the paper is structured as follows. Section 2 gives a brief overview of the related work in higher education students' motivation. Section 3 presents the set-up of our pedagogical experiment. Section 4 discusses experimental results. Conclusions and plans for the future work are given in Section 5.

2 Related Work

Motivations are denoted as "...reasons individuals have for behaving in a given manner in a given situation" (c.f. [3]). "They exist as part of one's goal structures, one's beliefs about what is important, and they determine whether or not one will engage in a given pursuit" (c.f. [4]). In academic settings two types of motivation are distinguished – intrinsic and extrinsic. Intrinsically motivated subjects learn for their own sake, because enjoy learning or assess the outcome of the learning process as important for themselves – e.g. [5]. Extrinsic motivation is driven by a desire of

getting rewards – from the others; or to avoid punishment. Students motivated extrinsically focus on receiving the approvals – like judgements by lecturers and peers – e.g. [4]. Our approach, though welcoming intrinsic motivation, focuses on obtaining utility of exploiting student’s extrinsic stimuli – which proves to become more spread and influential in the current economic settings.

Many authors stress the importance of a skill to maintain and enhance students’ motivation as one of the core capabilities of a University lecturer. “A wide variety of theories of learning and teaching recognises motivation as an essential prerequisite for successful learning. The ability to maintain and enhance student motivation is therefore one of the most important skills ..., and many publications and training programmes devote considerable space and time to this matter. Applying this theoretical knowledge in practice, however, remains difficult due to the complexity of the concept and the number of different models of motivation available” (c.f. [6]). Our research is focused exactly on the application of motivation stimuli to practice in a Bachelor level Computer Science programme – so the experimental data we have analysed spans across several disciplines taught in the 1-st to 4-th year of the programme at Zaporozhye National University (ZNU).

The mainstream of experimental studies in higher education teaching and learning is centred around using the methodologies of individual subjective assessment by subjects – based on interviews, questionnaires, etc (e.g. [7] to mention just one of many relevant publications). In difference to the mainstream methodology, we exploit the collaborative character that is intrinsic to student collectives and base our approach on well renowned social and peer approaches – this is why peer evaluation is used. Such a method allows us not only to collect and analyse individual judgements, but also to cross-rate the subjects by their own cross-judgements and stimulate healthy competition – thus increasing positive stimuli.

3 Setting up the Pedagogical Experiment

Stimulated by the necessity to seek for a remedy confronting the decrease of interest for quality learning at Universities, we have planned and further conducted a pedagogical experiment at ZNU. We have focused on the individual coursework as one of the important kinds of students’ creative activities in which motivation plays a very important role.

Our major objective was to prove a pedagogical hypothesis:

If students are given an opportunity to act as peer-evaluators of the other students reports, their **extrinsic motivation** to: (a) deliver the coursework; and (b) to perform as good as they can – **will be higher** than among those who do individual work in a traditional way and are graded by their instructor only. Furthermore, the **quality** of submitted reports is expected to **be better**, as students informally compete and cross-evaluate their quality. Finally, the **objectivity of the assessments will be higher**; those will be perceived as fair by the subjects.

For that we have:

- Chosen the disciplines: (a) for which the historical data on the coursework grades existed for several years; (b) for which the complexities of doing coursework assignments were comparable; and (c) the coverage of all four years of our Bachelor programme was even
- Developed detailed assessment forms for inexperienced evaluators offering a clear procedure and set of explicit metrics for coursework report assessment per each involved discipline
- Chosen the student groups comprising the cases when the same group acted as an experimental sample and formerly – a control sample; briefed the experimental groups
- Configured the set of software tools to support the experiment and developed written methodological recommendations for the subjects
- Adopted and adapted simple and effective metrics that allowed measuring the proofs of our research hypothesis

3.1 Pedagogical and Methodological Set-up

The pedagogical set-up of our experiment covers: the choice of disciplines; the preparation of the evaluation forms; and subjects' briefing about the evaluation procedure and tools.

First, we have chosen the disciplines with historical data and good coverage of our Bachelor programme. The choice is summarized in Table 1 – showing that:

- 3+ year historical data on the coursework assignment grades is available
- The disciplines cover all 4 years of study within the programme evenly

The complexity of the assignments, though different per discipline, is comparable as shown in Table 1.

- **Table 1.** Choice of disciplines and complexities of related coursework assignments.

Discipline	Year	2008	2009	2010	2011	Avg
Programming	I	---	100	100	100	100
Algorithms and Data Structures	II	100	150	200	250	175
DataBases and Information Systems	III	---	150	150	150	150
Intro to Logical Programming and AI	IV	---	150	150	250	183

Legend: numbers in columns 3-7 are coursework complexities. The cells corresponding to our experiment are shaded gray and have bolded numbers.

Grades data for the coursework assignments in Programming (year I) and Databases and Information Systems (year III) form our first and second baseline control datasets respectively.

The complexity of the 1-st year coursework assignment in Programming has been chosen as basic – represented by 100 abstract points. This coursework contains a survey part on a particular topic and a practical assignment to develop a program

solving a given simple problem. The complexity of the coursework in this discipline remains without change for all the 3 years of our observations.

The complexity of the 3-d year coursework in Databases and Information Systems is also static within the period of observation. However, it is 1.5 times more complex as contains several interrelated practical problems in database and IS development using SQL Server software. Another difference is that the subjects for this assignment were the III-d year students whose motivation from one hand and experience from the other hand differ from the ones of I-st year students.

Observations in Algorithms and Data Structures and Introduction to Logical Programming and Artificial Intelligence contain both control and experimental (shaded gray in Table 1) data.

The complexity of the coursework assignment in Algorithms and Data Structures increases from 100 points in 2008 to 250 points in 2011. In 2008 it was very similar in structure to the coursework in Programming – a detailed written presentation of a sorting algorithm studied individually and its practical implementation in a computer program. In 2009 the task of analytically evaluating the computational complexity of the algorithm was added – raising the complexity up to 150 points. In 2010 the task of experimental measurement of the computational complexity and comparing it to the analytical estimation was added – the complexity has therefore increased to 200 points. In 2011 the coursework has been complicated (up to 250 points) by offering a comparative evaluation exercise – the students were tasked to measure the performance of their program and compare to the performance of a program developed by a fellow based on several common datasets containing records of different types.

Secondly, we have developed the evaluation forms for coursework reports in both disciplines. An example of a fragment of an evaluation form is pictured in Fig. 1.

Reviewer:	X.Y.Zzzz		
Date:	DD.MM.2011		
Report No:			Overall Grade (0-15): 10.99
Section 1 Survey (0-6 points):			4.20
Basic Notions			
1.1:	Is the graph of the basic notions elaborated?		weight, % 20
<input type="checkbox"/>	1.00	yes - fully complies the definitions	grade 4.50
<input checked="" type="checkbox"/>	0.75	Insignificant mistakes	
<input type="checkbox"/>	0.50	Partial compliance to definitions	
<input type="checkbox"/>	0.25	Substantially incomplete or incorrect	
<input type="checkbox"/>	0.00	No	
	Please provide your arguments: Not all the required notions included: e.g. Markov Decision Process (MDP)		
1.2:	Are the basic terms denoted correctly and completely?		weight, % 20
<input checked="" type="checkbox"/>	1.00	Complete	grade 6.00
<input type="checkbox"/>	0.75	Insignificant omissions	
<input type="checkbox"/>	0.50	Important notions omitted	
<input type="checkbox"/>	0.25	Complete list without definitions	
<input type="checkbox"/>	0.00	No	
	Please provide your arguments: Major notions are given sufficiently completely		
<input checked="" type="checkbox"/>	1.00	Correct	weight, % 20
<input type="checkbox"/>	0.75	Insignificant mistakes	grade 6.00
<input type="checkbox"/>	0.50	Falsified sense	
<input type="checkbox"/>	0.25	Correct list without definitions	
<input type="checkbox"/>	0.00	No	
	Please provide your arguments: All definitions are correct		

Fig. 1. A fragment of the completed evaluation form for the coursework report in Introduction to Logic Programming and AI.

The forms are in fact structured questionnaires covering all the sections of the report and suggesting several weighted Likert scale [8] based metrics covering several aspects that were different for each section. Table 2 contains the lists of the report sections and evaluation questions for both disciplines.

Table 2. Evaluation aspects covered by review forms and scoring weights.

Algorithms and Data Structures		Intro to Logical Programming and AI	
Aspect to Evaluate	Weight %	Aspect to Evaluate	Weight %
Section 1: Sorting method and algorithm (0-2 points)		Section 1: Survey (0-6 points)	
1.1 Is the description of the family of sorting methods given?	25	1.1 Is the graph of the basic notions elaborated?	20
1.2 Is the algorithm described sufficiently completely and clearly?	50	1.2 Are the basic terms denoted correctly and completely?	40
1.3 Is algorithm stability analyzed?	25	Are the major problems in the field covered?	
Section 2: Software implementation (0-3 points)		1.3 Are the problem statements given?	10
2.1 Is the source code provided?	10	1.4 Is the actuality of these problems explained?	10
2.2 Does the implementation comply with the algorithm described in Section 1?	15	1.5 Are the descriptions of solution methods given?	10
2.3 Are the implementation decisions described sufficiently fully and clearly?	30	1.6 Are the surveyed solution methods compared?	10
2.4 Are the constraints (absence of) wrt input data explained and justified?	10	Section 2: Solving Problems in Visual PROLOG (0-7 points)	
2.5 Does the provided software work?	25	2.1 Is task 2 solved?	10
2.6 Is the source code reasonably commented?	10	2.2 Is task 3 (traffic expert system design) solved?	30
Section 3: Theoretical estimation of the computational complexity (0-2 points)		2.2.1 Are schematic descriptions of situations at T-shaped crossroad given?	
3.1 Is the estimation the computational complexity given?	50	2.2.2 Are predicates and goals for situations fully described?	
3.2 Is the graphical illustration of the computational complexity given?	50	2.2.3 Are predicates and goals compliant to the schematic descriptions?	
Section 4: Computational experiment – comparison with other algorithms (0-3 points)		2.3 Is task 4 (traffic expert system implementation) solved?	30
4.1 Are the data sets chosen correctly	50	2.3.1 Are predicates and goals specified correctly?	
4.2 Are the other algorithms for comparison chosen reasonably?	50	2.3.2 Does the developed expert system work?	
Section 5: Experimental assessment of the computational complexity (0-3 points)		2.3.3 Are the implementation decisions documented?	
5.1 Are the rules and programming solution for measuring computational complexity described?	30	2.4 Is task 5 (traffic expert system refinement) solved?	30
5.2 Is the comparative analysis of the computational complexity given?	30	2.4.1 Are predicates and goals specified correctly?	
5.3 Is the experimental assessment compared with the theoretical estimation?	30	2.4.2 Is the sense of the predicates and structures explained?	
5.4 Is the graphical illustration of the computational experiment results given?	10	2.4.3 Are all possible traffic situations described?	
Section 6: Conclusions (0-3 points)		2.4.4 Does the refined Expert System work?	
6.1 Do the conclusions reflect the results obtained?	50	Section 3: Conclusions (0-2 points)	
6.2 Are the references to the adopted components given?	30	3.1 Do the conclusions reflect the results obtained?	50
6.3 Is the report compliant to the template (abstract, contents, references, appendices)?	20	3.2 Are the references to the adopted components given?	30
		3.3 Is the report compliant to the template (abstract, contents, references, appendices)?	20

It has been decided that the overall grade for a coursework report of maximum 20 points is divided in two parts:

- The coursework grade (0 – 15 points) computed as a mean of the three assessments done by two peers and one instructor
- The evaluation grade (0 – 5 points) computed as 5 minus the mean of deviations of the subject's evaluation scores from the mean scores. So, the closer an individual scoring is to the mean scoring in all the evaluation assignments – the higher the resulting evaluation grade is.

3.2 Experimental and Control Groups

Two experimental groups in the II-nd and IV-th year of study have been selected so that the historical coursework grade data was available for them. For comparison, the control data about the grades in the other groups of different years of study and in all four chosen disciplines have been taken into account. The groups for which the control data was accounted for have been further treated as control groups. Table 3 depicts the distribution of the control and experimental groups over the years of study. As could be seen in Table 3, the experimental groups are also control groups but in different disciplines and years of study. So, different ways of comparing the activity and performance in doing coursework assignments arise: the same group in different years; the same group in different disciplines; the same group as experimental and doing the work in a traditional way; etc.

Table 3. Experimental and control groups.

Group No	2008		2009		2010		2011	
	Year of Study	Role	Year of Study	Role	Year of Study	Role	Year of Study	Role
8216			IV	C				
4327	II	C	III	C	IV	C		
4328			II	C	III	C	IV	E
4329			I	C	II	C	III	C
4320					I	C	II	E
4321							I	C

Legend: C – control group; E – experimental group.

At the beginning of the experiment the subjects of our two experimental groups were briefed about: the deadlines; the objectives of peer evaluation; the structure and the content of the evaluation forms; the grades that will be assigned for the reports and for the reviews; the tools they will use in the peer evaluation process.

3.3 Instrumental Set-up

Two procedures have been chosen for evaluation that differed in the used tools. For the experiment with the II-nd year students the workflow based on e-mail exchange and manual supervision has been adopted. For the IV-th year students we have

introduced the EasyChair Conference Management System¹ as a tool to manage the process, final ranking and grading. In both cases the structured evaluation forms have been offered to the subjects to be filled out using MS Excel.

4 Results and Discussion

Evaluation process has been organized and executed similarly to peer evaluation of conference papers by programme committee members. Students were invited to serve on the programme committee and the review assignments have been made by the instructors who acted as programme chairs. The results of evaluation have been collected and processed using two different patterns:

- For II-nd year students – collected by e-mail and processed manually using Excel spread sheet as shown in Fig.2 in an anonymized way

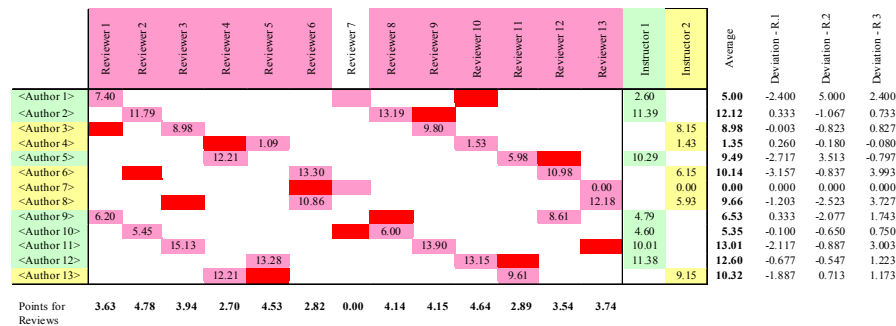


Fig. 2. Anonymized review results visible to instructors.

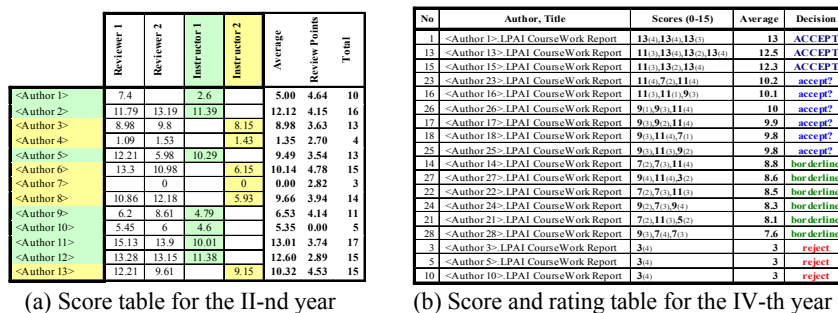


Fig. 3. Resulting score and rating tables communicated to the students.

- For IV-th year students – collected and processed using the EasyChair installation resulting in a very similar score table as the one in Fig. 2

The authors of the coursework reports have been further notified by e-mail about their results and ratings in the overall list as pictured in Fig. 3. Again, the notifications

¹ <http://www.easychair.org/>

to the II-nd year students have been manually communicated by e-mail; and the IV-th year students have been notified by the EasyChair.

4.1 Additional Effort for Tutors

As experienced, the additional instructors' effort for organizing and managing the peer review process was substantial.

The major part of their additional work could be qualified as the set-up effort: developing review forms; creating review environments; compiling briefing manuals for the student reviewers; preparing management tables; and configuring the software tools. The result of this effort may however be re-used quite substantially – so the start-up effort may be regarded as an initial investment and neglected in further considerations.

Following two different workflows for the II-nd and IV-th year students implied different management efforts because of using different toolsets. Overall, using EasyChair Conference Management System appeared to be about 3 times less effort consuming than using just e-mail and MS Excel.

4.2 Interpretation of Experimental Results

Table 4 contains the summary of our experimental findings and is structured as follows:

Table 4. Experimental Results.

Discipline	Year		Group No	Avg Score (0-20)	Avg Submission Ratio			Avg Score among Submitted	
	of Study	Calendar			No Submissions	Total Students	Ratio	Factual (0-20)	Aligned by Complexity
Programming (PR)	I	2009	4329	5,00	9	23	0,39	12,78	12,78
		2010	4320	10,33	9	15	0,60	17,22	17,22
		2011	4321	4,06	7	16	0,44	9,29	9,29
Algorithms and Data Structures (ADS)	II	2008	4327	9,68	21	31	0,68	14,29	14,29
		2009	4328	12,03	21	29	0,72	16,62	24,93
		2010	4329	8,50	10	19	0,53	15,30	30,60
		2011	4320	10,13	13	15	0,87	11,69	29,23
DataBases and Information Systems (DBIS)	II	2009	4327	11,70	14	23	0,61	19,21	28,82
		2010	4328	12,00	18	33	0,55	18,67	28,00
		2011	4329	11,00	12	19	0,63	18,33	27,50
Introduction to Logic Programming and AI (LPAI)	IV	2009	8216	4,47	10	36	0,28	16,10	24,15
		2010	4327	4,76	6	21	0,29	16,67	25,00
		2011	4328	7,14	15	28	0,54	13,33	33,33

- Broad horizontal sections correspond to the data related to one discipline. Two of them are baseline (as explained in Section 3.1) – Programming and Databases and Information Systems. The other two contain both control and experimental data – Algorithms and Data Structures (II-nd year) and Introduction to Logic Programming and AI (IV-th year).
- The Year column informs about the timing attribution of data (years of study and calendar years);
- The Group No column associates the rows to the academic groups. Group numbers may be found similar in several cases – reflecting the availability of both control and experimental measurements for several groups in different years and disciplines.
- The average scores are in fact based on the total number of students in a group which makes it different to the scores in the last two columns computed based on the number of submitted reports.
- Average Submission Ratio is in fact the measure that reflects the motivation of our students to submit their work
- The Factual Average Scores are the averages for the submitted reports, but without balancing them by coursework complexity
- Finally, the rightmost column contains the score averages multiplied by the complexity scaling factors provided in Table 1

Let us explain now how the results given in Table 4 and further interpreted graphically in Fig. 4 prove our research hypothesis.

Firstly, we expected that the introduction of peer reviews as an untraditional way of teaching will increase students' **extrinsic motivation**. This expectation was valid as pictured by the values of submission ratio. Indeed, the ratio of coursework submission in our experiment with the II-nd year students reached the global maximum of 0.87 across all the disciplines. The next lower value was 0.72 which is 15 per cent lower. For the IV-th year subjects the increase in motivation was not that significantly high overall, though very substantial within their year of study. Indeed the reached submission ratio of 0.54 is 1.86 times better than the next lower value of 0.29 in 2010.

Secondly, the quality of submitted reports may have been interpreted as quite average in our experiments: 11.69 in the II-nd year and 13.33 in the IV-th. The registered decrease in scores, compared to the previous year, is: 23.96 per cent for the the II-nd year; and 20.03 per cent for the IV-th year. A compensation for that decrease in quality is twofold:

- (i) As the ratio of submissions increased the proportion of the best students (who always submit their work) decreased – so did the average scores. For the II-nd year the ratio increase was 15 percent versus a 23.96 decrease in scores. However, for the IV-th year the increase in submission ratio (86 per cent) substantially outperformed the decrease in average score (20.03 per cent). So, it could be concluded that our approach proved to be effective for the final year students of our Bachelor programme.
- (ii) The observed decrease in scores is to some extent explained by the increase of coursework complexity. Indeed, the maximal values of the average scores have been reached in the cases with substantially less complicated coursework assignments – as explained in Table 1. For example, the global maximum of 19.21

corresponds to the assignment weighted 150 points. It is ‘outperformed’ by the score of 13.33 in our IV-th year experiment because the complexity of the experimental coursework is 250 points. This imbalance is corrected by the values shown in the Aligned by Complexity column of Table 4.

Figure 4 pictures the trends observed in our experiment graphically. The Y-values in Fig. 4(a) are the numbers from the Submission Ratio column of Table 4; while the Y-values in Fig. 4(b) are taken from the Aligned by Complexity column of this table.

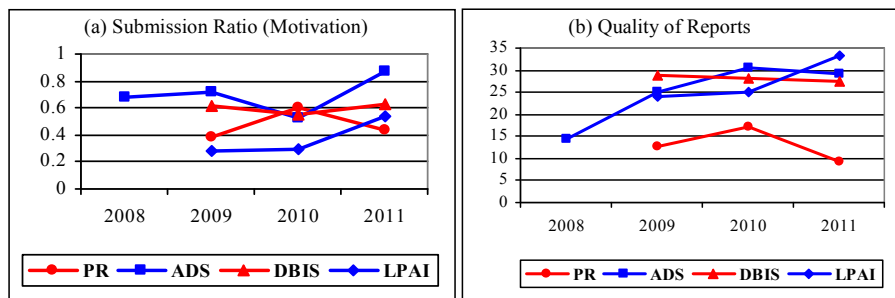


Fig. 4. Graphical interpretation of the increase in motivation and quality of work.

Finally, we hypothesized that the **objectivity of the assessments will be higher** in our experiment. We did not elaborate a proof for that as we did not undertake an experiment for scientifically measuring the objectivity. However, as a very draft estimation, we interviewed our subjects informally. These interviews revealed that the students treat their scores as more clear and objective compared to the previous experience, even if the scores were lower both individually and on average (column Factual of Table 4).

5 Conclusions and Outlook

This paper reported about our pedagogical experiment undertaken for seeking a way of improving extrinsic motivation and learning quality of Computer Science students in our Bachelor programme at ZNU. The increase in motivation has been proven convincingly. Exploiting students’ aspirations for informal leadership and incurred competition constructively is effective and attracts people to learning. A gain in the quality of learning was a little bit over-estimated. Indeed, having involved more students in a creative learning activity does not guarantee that the quality of their work increases dramatically by miracle. However, the increase in motivation helped increasing also the quality to some degree – as shown in the previous section.

A good side effect is also that the students learn the working patterns of the professionals in their field broadly used in academia and industry for making qualitative and unbiased peer evaluations.

The results discussed in Section 4 appeared to be positive also for the other colleagues at the department of IT at our University. So, we plan to extend the experiment by covering more disciplines and collecting a broader sample of results in

the near future. Among other things, this will allow us basing our work on a statistically representative set of subjects and making our results statistically valid. Finally, we plan to undertake an evaluation of the objectivity of the scoring in our settings.

References

1. Bound, J., Lovenheim, M.F., Turner, S.: Why Have College Completion Rates Declined? An Analysis of Changing Student Preparation and Collegiate Resources. *Am Econ J. Appl. Econ.* 2(3), 129--157 (2010), doi: 10.1257/app.2.3.129
2. Clark, P. B., Wilson, J. Q.: Incentive Systems: A Theory of Organizations. *Administrative Science Quarterly*, 6(2), 129--166 (1961)
3. Middleton, J. A., Photini A. Spanias: Motivation for Achievement in Mathematics: Findings, Generalizations, and Criticisms of the Research. *J. Research in Mathematics Education*, 30(1), 65--88 (1999)
4. Ames, C.: Classrooms: Goals, structures, and student motivation. *J. Educational Psychology*, 84, 261--271 (1992)
5. Middleton, J. A.: A study of intrinsic motivation in the mathematics classroom: A personal constructs approach. *J. Research in Mathematics Education*, 26, 254--279 (1995)
6. Ramirez-Iniguez, R., Canton, U.: Understanding Motivation in Large Groups of Engineering and Computing Students. In: *Inspiring the next generation of engineers. Proc. Engineering Education* (2010)
7. Wong, S.H.S: Motivating students to learn through good and helpful coursework feedback. In: *Inspiring the next generation of engineers. In: Inspiring the next generation of engineers. Proc. Engineering Education* (2010)
8. Likert, R.: A Technique for the Measurement of Attitudes. *Archives of Psychology*, 140, 1--55 (1932)

Approach to E-Learning Fundamental Aspects of Software Engineering

Ekaterina Lavrischeva¹, Alexei Ostrovski¹, and Igor Radetskiy¹

¹Institute of Software Systems of NAS, Akedemika Glushkova str., 40, Kiev, Ukraine
{lavryscheva, ostrovski.alex}@gmail.com, iradetskiy@mail.ru

Abstract: New theoretical and applied aspects of software engineering are introduced, viz.: technologies of developing programs and reusable components with MS.NET, CORBA, Java, Eclipse environments; assembling them into applied systems and their families; embedding components into the modern environments for shared usage; modeling applied domains in ontological DSL-like languages with tools like MS DSL Tools, Workflow, Eclipse-DSL, and Protégé. These aspects are implemented in the instrumental and technological complex (ITC). They are oriented towards improving software industry based on the readymade software resources (reuses, assets, services, artifacts). The ITC is represented by a web site with modern design, the contents of which has no known counterparts. The site is introduced as a tool for developing various kinds of programs and systems in the corresponding product lines, as well as for teaching computer science students the subject of software engineering.

Keywords: software engineering, interoperability, programming methodology, software industry, e-learning SE.

Key terms: Model, Process, Management, Environment, Development.

1 Introduction

The fundamental project III-1-07 of NAS of Ukraine “Theoretical Fundament of Generative Programming and Means of Its Support” (2007-2011) paved the road to several new methods of developing complex programs from more simple software resources. They have been created at the software engineering department at Institute of Software Systems, Ukrainian National Academy of Sciences.

The defined task was to develop new scientific and applied aspects of software engineering, directed towards the advances in assembling software products from the readymade software resources (reuses, aspects, services, etc.). To fulfill this task, we studied and took into account the modern facilities and advances in the domain of software engineering, such as object-component programming, generative, assembling, agent-oriented and service-oriented programming [1-5], as well as peculiarities of modern operating environments and systems (Microsoft .NET,

CORBA, Java, IBM, Eclipse, Protégé and others). This was done in order to implement the industrial aspects of software engineering on the basis of the reuse technique. As a result of efforts in the scope of the aforementioned fundamental project, a new theoretical foundation in producing applied systems (AS) and software product families have been developed and the previously known one has been substantially improved. Most of these efforts have found an implementation in the created instrumental and technological complex (ITC).

2 New Aspects in Software Engineering

Research and designing within the III-1-07 project were conducted in the three main directions (Fig. 1).

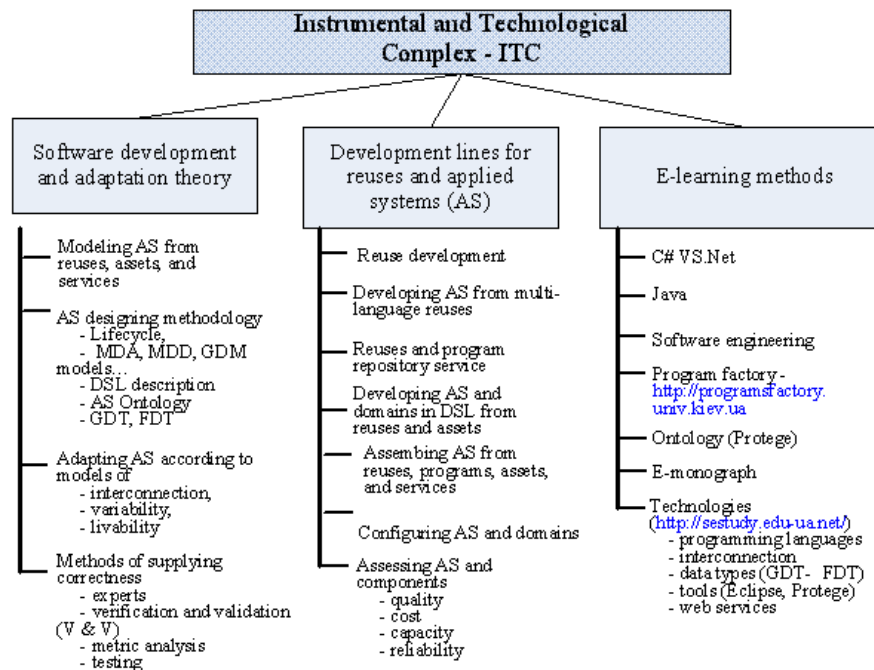


Fig. 6. Structure of the ITC.

The scientifically important results of the long-term research and development are briefly listed below (their full descriptions may be found in the electronic paper [6] and other published works):

1. Interfaces and techniques for assembling complex programs from reusable components and software resources [2], [3], [7-10].

2. The new object- and component-oriented approaches to the development of software product families [10], [12].
3. The methodology of producing programs and systems on the basis of the subject-oriented domain-specific language (DSL) and software resources, particularly reusable components, assets and services [1], [7], [12-15].
4. The theories of interoperability, variability (i.e., adaptability and volatility of software product families), persistence capability, and fault tolerance in programs and software systems [16-19].
5. The concepts of program factories and technological product lines for software development [1-6], [9], [20-23].
6. The quality assessment for applied systems and lifecycle processes [6], [11], [16].
7. The method of representing information on software resources and reusable components in the program repository [14], [16].
8. The techniques to work with the product lines and the tools, such as Protégé, Eclipse, CORBA, Eclipse-DSL, Ant, C#, Java, Basic within the ITC [6], [24], [25].
9. Studies, textbooks and manuals on software engineering [3-5], [15].

The listed scientific advances are mostly implemented within the ITC, which is oriented in assisting the development of applied systems and software product families from reusable components on the basis of generic product lines [6], [16]. The operations with the components are provided by the web site <http://sestudy.edu-ua.net> and by the program factory for Kiev National University students (<http://programsfactory.univ.kiev.ua>), which are both implemented under the supervision of Prof. E.M. Lavrischeva.

Judging by the list of several criteria, such as the wide coverage of various software engineering topics and supplying product lines with detailed descriptions and examples, the ITC has no known counterparts with free access in the ex-USSR region of the Internet. The introduced modular architecture of the complex allows widening its functionality with ease by adding new product lines or elaborating the existing ones.

3 Functions and Structure of the Web Site

The site in question was developed as a collection of tools for software engineering and at the same time was displayed during lectures on software engineering at Kiev National University. This drove authors to orient the complex towards teaching students and graduate students the basics of software engineering, including various tools and means of their support. The following main aspects of SE were singled out: assembling software systems and their families on the basis of software resources and reusable components, techniques for developing, generating, interoperability, and ontological modeling of the object domains. Besides that, electronic technologies of software development in programming languages like Java, C#, C++, Basic, and others were included into the course.

Taking the above into account, we have chosen a strategy of teaching various aspects of industry-compliant software engineering. In order to gradually and consistently implement this strategy within the ITC, we utilized the Internet-based

methods and modern programming systems that support different aspects of software development, namely:












- Protégé system for modeling object domain ontologies.
- Eclipse as a tool to embed different programming and system components into the ITC by using its plug-ins.
- Microsoft Visual Studio .NET as a multifunctional tool to organize team development of the new systems, including developing software via Internet using various programming languages, OOP, UML, and cloud computing frameworks, such as Azure, SkyDriven, Amazon, etc.
- CORBA system that has a universal broker providing interoperability between programs, written in different languages, by using the time-proved stub/skeleton mechanism.
- New subject-oriented DSL tools with a graphical user interface for designing systems, domains, applications, and families, and for implementing DSL-based descriptions, i.e. Eclipse-DSL, Microsoft DSL Tools, and others.





The start page of the web site features a list of implemented sections and subsections concerning software engineering. The sections in question are: Main Page, Technologies, Interoperability, Tools, Presentations, and Learning (Fig. 2). Each section contains subsections with keywords that specify the names of product lines (17 altogether). All sections and subsections include standardized pages, such as an overall theoretical description, an example that illustrates the concerned topic (developed with one of the workbench programming environments, in most cases), a thorough description of the example, and so on.




During the course of choosing product lines for their inclusion into the complex, the following main criteria were taken into account (with the order reflecting their priority, from the most important to the least significant ones):


5. Relevance of the topic in question within the software engineering discipline, as well as its applicability for solving present-day real-world problems.
6. Theoretical basis supporting the technology.
7. Relations between the technology and techniques behind other product lines.
8. Simplicity and accessibility of both theoretical and applied aspects of the technology for a sufficiently wide audience, including students and lecturers of Ukrainian universities.
9. Availability of an illustrative example to explain the main concepts behind the technology in question, preferably with applicability to certain real-world problems.

Employees of the software engineering department and students of KNU and MIPT implemented the web site and several product lines of software development on the basis of readymade resources and components in course of writing their thematic and graduate papers. Particularly, they have developed an experimental program factory, software means of interoperability support between programs and systems, a domain description in DSL using Protégé environment, and a system for registering academic missions in the institutes of NAS of Ukraine.

-  Main Page
-  Techniques
 -  Reuse Repository
 -  Reuse Development
 -  Reuse Assembling
 -  Reuse Configuration
 -  Generating DSL descriptions
 -  Quality Engineering
 -  Ontology (lifecycle, geometry)
 -  Web-services
 -  GDT—FDT Transformation

-  System Interoperability
 -  CORBA—Eclipse
 -  VS.NET—Eclipse
 -  Visual Basic—Visual C++

-  Instruments
 -  Eclipse
 -  Protégé

-  Presentations
 - Applied System
 - Software Engineering and Factories
 - Software Industry


-  Learning
 - C# and MS.NET
 - Java
 - Software Engineering

Fig. 7. Keywords in the main page of the web site

4 Technical Implementation of the Web Site

Because of several technical issues, using traditional content management systems (CMS) in the implementation was deemed impossible, or, at least, vastly complicated. Due to this, the chosen architecture for the content representation is an interim

solution between static web pages and the acknowledged Model—View—Controller (MVC) architecture.

Each page displaying an article on one of the topics of the complex is built using the same template that contains the following main components:

- The unified header, which contains the site banner and the title.
- The current location string.
- The main menu including the language panel and links for navigation.
- The navigation panel, which contains links to various subsections of the current section.
- The content of the article.
- The footer that includes information about the site authors and developers.

Creating the dynamic page components (all of the aforementioned ones, except for the header and the footer) is done using PHP programming language. The utilized tree structure for representing sections, subsections, and corresponding articles allows generating these components with ease. The SQLite database is used as a persistent storage for the data, such as article titles and contents, due to the eliminated need in the dedicated server process. As some similar structures in articles (for example, numbered figures and information about downloads) are frequently reused, the content of articles may include not only standard HTML tags, but also custom XML tags, which are translated into HTML code by a simple preprocessor.

5 Description of the Web Site Contents

5.1 Main Page

The main page contains the description of the subject of software engineering according to the corresponding body of knowledge — SWEBOK, which was developed in 2001 by the international committee, formed by ACM and IEEE. The body of knowledge consists of ten areas of knowledge and gives the following definition of software engineering:

Software engineering is a system of methods, means and disciplines for designing, developing, running and supporting software.

However, SWEBOK does not give a sufficient description of software production and quality (for example, it lacks languages for describing specific domains, theories of project decision analysis, data protection, product lines, software documenting, etc.). It also does not provide software development, control, and economy disciplines. In order to overcome these complications, we propose a new concept for breaking down the software engineering disciplines (Fig. 3):

- Scientific discipline consists of the classic sciences (theory of algorithms, set theory, logic theory, proofs, and so on), lifecycle standards, theory of integration, theory of programming and the corresponding language tools for creating abstract models and architectures of the specified objects, etc.

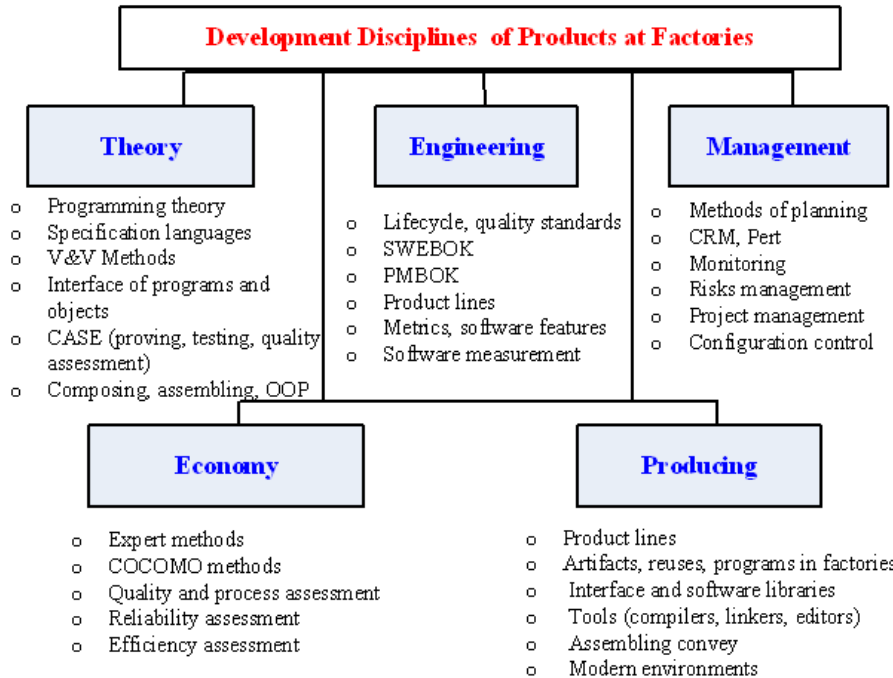


Fig. 8. Classification of software engineering disciplines.

- Engineering discipline is a set of technical means and methods for software development by using standard lifecycle models; software analysis methods; requirement, application and domain engineering with the help of product lines; software support, modification and adaptation to other platforms and environments.
- Management discipline contains the generic management theory, adapted to team-based software development, including job schedules and their supervising, risk management, software versioning and support.
- Economy discipline is a collection of the expert, qualitative and quantitative evaluation techniques of the interim artifacts and the final result of product lines, and the economic methods of calculating duration, size, efforts, and cost of software development.
- Product discipline consists of product lines, utilizing software resources (reusable components, services, aspects, agents, and so on), taken from libraries and software repositories; it also contains assembling, configuring and assessing quality of software.

These disciplines are built on the basis of software engineering [13], modern approaches, and the scientific fundament; they are used in developing product lines and estimating quality of software products.

5.2 Technologies

The developed embedded technology to work with software products is represented by the following list of product lines for software component development in the ITC:

- The program factory, which contains the specification of reusable components and courses on basic MS .NET programming and software engineering, and the students' program factory, developed at the cybernetics department of Kiev National University, as an example of such a factory.
- The repository of reusable components, which is an integral part of the aforementioned factory.
- Assembling multi-language programs and components into a software system by converting incompatible data types.
- Configuring reusable components in a system with complex structure that possesses points of possible modifications in some subprograms by the customer's wish (so called variability points), designed with MS .NET Workflow environment.
- Describing applied domains in DSL by example of the lifecycle domain (IO/IEC 12207-2007 standard) with graphical and textual representations, created with Eclipse-DSL environment.
- Quality and cost engineering with the help of softest application, designed to estimate labor expenditures and the cost of software development.
- Designing domain ontology by example of the applied domain of computational geometry with Protégé environment.
- Constructing software product families by merging components that use different programming platforms with the help of web services.
- Translating general and fundamental data types (GDT and FDT) according to ISO/IEC 11404 standard and GRID system programming practices, by example of the primitive library.
- Generating software resources and merging them into programs, software products, and their families with the configurator, as specified by the variability model.
- Testing programs in order to obtain a correct software product and to collect data about faults and errors, required in assessing its operational reliability.

In general, implementing the new product lines for the gradual development of software products by merging generic lines with the help of new methods lets us conclude that software engineering is approaching the needs of modern program factories.

5.3 Interoperability

The Internet nowadays supplies various forms of interaction and interoperability between distributed systems, environments, and their tools.

Interoperability between programs, systems, and environments in the ITC is developed according to a new interaction theory [12], [18-22]. The goal of the theory can be briefly summarized as improvement of the common access methods to provide the software portability between programming environments residing within the common ITC repository (Fig. 4).

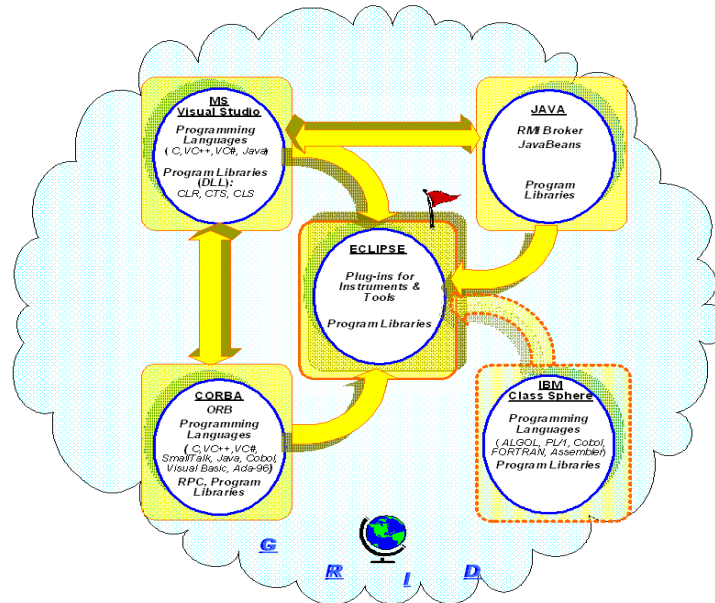


Fig. 9. Structure of interconnection between general environments.

These operational environments support lifecycle processes for developing heterogeneous programs and merging them into various software structures by using specific connection mechanisms. The implemented interoperability techniques have no real theoretical counterparts; they are rather tested in practice with the listed examples:

1. Interoperability between programs created in Visual Basic and Visual C++, provided by the interface layer in form of a library, which transmits data from one program to another and transforms incompatible data types, when necessary.
2. Interoperability between Java and Microsoft .NET programming platforms, implemented by utilizing the CORBA object request broker and using interface definition language (IDL) to describe interfaces in these platforms.
3. Interoperability between Microsoft Visual Studio and Eclipse integrated development environments, provided by transmitting application data of a program, developed with Visual Studio, into the Eclipse repository, utilizing Eclipse plug-in capabilities.

5.4 Tools

The section contains the description of Eclipse IDE and its use in merging various workbench tools by utilizing its capabilities to widen functionality with the help of plug-ins. The second development environment included is Protégé, which is used to create the models of applied domains and then to represent them in the modern subject-oriented DSL language. The considered examples are: creating reusable component repository in Eclipse in order to develop new applications, and developing

the ontological model of informational and technical resources from the Internet with the help of Protégé.

5.5 Presentations

The section in question contains the three following presentations on the subject of software engineering:

1. The automated system of production activities of the foreign affairs department of Ukrainian National Academy of Sciences.
2. The fundamental principles in designing program factories, structures, software resources and component repositories, methods and tools to support development in processing lines.
3. The concept and aspects of software industry, proposed in the previously mentioned fundamental project.

5.6 Learning

The Learning section consists of the three processing lines:

1. Distance learning of modern programming languages and environments, namely C# and MS Visual Studio.
2. Learning Java programming language with the textbook by I. Khabibullin (St.-Petersburg) that is freely available and contains numerous examples of programming and translating processes, and program execution.
3. Learning software engineering with the electronic textbook by Prof. E. Lavrischeva, which is available both in Ukrainian and in Russian.

6 Conclusions

The instrumental and technological complex is developed as a web site in the corporate network of the Institute of Software Systems in order to support software production with the simplified general-purpose product lines. It implements the following concepts and methods:

- Organizing interoperability in heterogeneous software by utilizing the introduced interaction model for programs and systems that can be transferred into a different environment and executed with data transmitted through interfaces or acquired from databases and modern online data stores, such as SkyDriven.
- Technologies of reusable components' development and describing interface data according to WSDL standards; storing interfaces and components in the repository; using the repository to provide a reliable source for readymade software components to be used by third-party developers in engineering new systems.
- Assembling heterogeneous programs from the available reusable components working under different programming platforms, which possess passport data required to merge components and to translate incompatible transmitted data types.
- Describing domains of complex systems (the lifecycle domain of ISO/IEC 12207 standard, computational geometry, software testing) in DSL and implementing

them with Visual Studio .NET DSL Tools or Eclipse-DSL, including an example of the testing process.

- Generating primitive transformation functions for several data types (table, array, sequence, etc.) between GDT and FDT according to ISO/IEC 11404 standard.
- Configuring various bits of source code and components from software product families according to a generic variability model.
- Learning software development in C#, Java, Basic programming languages with VS.NET and Eclipse environments, as well as software engineering with the e-textbook.

Examples that demonstrate the listed technologies from the complex are implemented with the help of its development environments (Eclipse, Visual Studio .NET, and so on). They meet several generic criteria for developing applications such as correctness, soundness, and intuitive design.

The main prospective lines of development are as follows:

- Finalizing methods of resources composition with the help of services followed by configuring, verifying or testing the readymade resources and applied systems.
- Improving the quality model of software product sets for the class of critical systems; completing it with reliability models based on data on intensity of program faults and assessed variability points, which can influence quantitative evaluation of software product quality (these models may utilize Bayesian networks or trees).
- Improving the concept of component certification in terms of compliance with the generally accepted standards and adequately imposed requirements on software.
- Continuing developing the web site by adding new software engineering disciplines and computer science topics with possibility of distance learning, which may help widen the circle of its users (primarily, students and lecturers from Ukrainian universities).

References

1. Czarnecki, K., Eisenecker, U.: *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley, Boston, MA (2000)
2. Lavrischeva, K.: *Compositional Programming: Theory and Practice*. In: *Cybernetics and Systems Analysis*, vol. 45, no. 6, pp. 845–853. Springer, Heidelberg (2009)
3. Lavrischeva, E., Grischenko, V.: *Assembly Programming. Basics of Software Industry*. Naukova Dumka, Kiev, 2nd ed. (2009) (in Russian)
4. Bai, Y.: *Applications Interface Programming Using Multiple Languages: A Windows' Programmer's Guide*. Prentice Hall Professional, Upper Saddle River (2003)
5. Greenfield, J., Short, K., Cook, S., Kent, S.: *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*. Wiley, Hoboken (2004)
6. Lavrischeva, E., Koval, G., Babenko, L., Slabospitska, O., Ignatenko, P.: *New Theoretical Foundations of Production Methods of Software Systems in Generative Programming Context*. Electronic monograph, in: UK-2011, vol. 67. VINITI RAN, Kiev, Moscow (2012) (in Ukrainian)
7. Lavrischeva, E.: *Generative Programming of Software Products and Their Families*. In: *Problems in Programming*, vol. 1, and pp. 3–16. Akadempriodika, Kiev (2009) (in Ukrainian)

8. Lavrisheva, K.: Formal Fundamentals of Component Interoperability in Programming. In: Cybernetics and Systems Analysis, vol. 46, no. 4, pp. 639–652. Springer, Heidelberg (2010)
9. Lavrisheva, E.: Formation and Development of the Modular-Component Software Engineering in Ukraine. Institute of Cybernetics after V. Glushkov, Kiev (2008) (in Russian)
10. Grischenko, V.: Object-Component Designing Method for Software Systems. In: Problems in Programming, vol. 2, and pp. 113–125. Akademperiodika, Kiev (2007) (in Ukrainian)
11. Andon, P., Koval, G., Korotun, T., Lavrisheva, E., Suslov, V.: Foundation of Quality Engineering of Software Systems. Akademperiodika, Kiev, 2nd ed. (2007) (in Russian)
12. Lavrisheva, E.: Problem of Interoperability between Heterogeneous Objects, Components, and Systems. Approach to Solve It. In: 7th International Programming Conference “UkrProg ‘2010”, pp. 28–41. Akademperiodika, Kiev (2010) (in Russian)
13. Lavrisheva, E.: Classification of Software Engineering Disciplines. In: Cybernetics and Systems Analysis, vol. 44, no. 6, pp. 791–796. Springer, Heidelberg (2008)
14. Lavrisheva, E., Slabospitska, O.: An Approach to Expert Assessment in Software Engineering. In: Cybernetics and Systems Analysis, vol. 45, no. 4, pp. 638–654. Springer, Heidelberg (2009)
15. Lavrisheva, E.: Software Engineering. Textbook. Akademperiodika, Kiev (2008) (in Ukrainian)
16. Lavrisheva, E., Slabospitska, O., Koval, G., Kolesnik, A.: Theoretical Aspects of Variability Management in Software Product Families. In: KNU Bulletin, Physics and Mathematics Series, vol. 1, pp. 151–158. KNU, Kiev (2011) (in Ukrainian)
17. Lavrisheva, E.: Interaction Models of Programs, Systems, and Operational Environments. In: Problems in Programming, vol. 3, pp. 13–24. Akademperiodika, Kiev (2011) (in Ukrainian)
18. Ostrovski, A.: Approach to Interconnection Support between Java and MS.NET Programming Environments. In: Problems in Programming, vol. 2, and pp. 37–44. Akademperiodika, Kiev (2011) (in Russian)
19. Radetskyi, I.: One of Approaches to Maintenance Interconnection Environments Visual Studio and Eclipse. In: Problems in Programming, vol. 2, and pp. 45–52. Akademperiodika, Kiev (2011) (in Ukrainian)
20. Aronov, A., Dzubenko, A.: Approach to Development of the Students’ Program Factory. In: Problems in Programming, vol. 3, and pp. 42–49. Akademperiodika, Kiev (2011) (in Ukrainian)
21. Lavrisheva, E.: Concept of Scientific Software Industry and Approach to Calculation of Scientific Problems. In: Problems in Programming, vol. 1, and pp. 3–17. Akademperiodika, Kiev (2011) (in Ukrainian)
22. Andon, P., Lavrisheva, E.: Development of Program Factories in the Informational World. In: Bulletin of NAS of Ukraine, vol. 10, and pp. 15–41. Akademperiodika, Kiev (2010) (in Ukrainian)
23. Lavrisheva, E.: Theoretical and Applied Aspects of Software Systems Development. In: TAAPSD’2010, pp. 274–285. Kiev (2010) (in Ukrainian)
24. Lavrisheva, E.: Instrumental and Technological Complex for Developing and Learning Aspects of Software System Development. In: Bulletin of NAS of Ukraine, vol. 3, and pp. 67–79. Akademperiodika, Kiev (2012) (in Ukrainian)
25. Anisimov, A., Lavrisheva, E., Shevchenko, V.: On Scientific Software Industry. Technical report, Conf. Theoretical and Applied Aspects of Cybernetics (2011) (in Ukrainian)

Choosing the First Educational Programming Language

Vladyslav Kruglyk¹ and Michael Lvov¹

¹ Kherson State University, 40 Rokiv Zhovtnya, 27
73000, Kherson, Ukraine
kruglik@ksu.k.s.u.a, lvov@ksu.k.s.u.a

Abstract. The article describes requirements to educational programming languages and considers the use of Python as the first programming language. The issues of introduction of this programming language into teaching and replacing Pascal by Python are examined. The advantages of such approach are regarded. The comparison of popular programming languages is represented from the point of view of their convenience of use for teaching algorithmization and programming.

Keywords: Educational programming language, Python, Pascal, first programming language.

Key Terms: Didactics, CompetenceFormationProcess, InformationCommunicationTechnology, TeachingMethodology

1 Introduction

The informatization of society lays new claims to teaching programming. Information technologies are strategic area of economy development in Ukraine, and the significance of their development and support is equal with the development of electronics in the last century.

At the end of the last century humanities have become very popular specialties. A school course of informatics was altered as well. As for now, a programming course has been almost totally excluded of secondary schools curriculum. The subject “Informatics” has been focused on teaching office technologies, and the proprietary software of one famous company was chosen as an example (such choice has been made historically). Therefore, the restoration of students’ background in mathematics and informatics has become an urgent and important issue.

1.1 Overview of Current Situation

At schools conceptually different methodological approaches are used to teach programming languages. They are: equipped and non-equipped. The essence of the first one is in teaching programming without computers, by using so-called

educational algorithmic languages. From our point of view, such approach was reasoned by the only factor - a lack of computers at schools.

The second approach implies teaching the basics of algorithmization and programming at school with the use of industrial programming systems. Mostly BASIC, TURBO PASCAL and visual programming environments, like DELPHI and VISUAL BASIC on the basis of BASIC and OBJECT PASCAL are used as programming languages at schools.

The programming languages, named above, are developed for industrial purposes, and they are not intended for teaching, because they don't explicitly contain the means, based on the concepts of algorithmization and programming. Let's remember words of the academician A.P. Ershov, who asserted that educational programming language has to involve all main programming concepts [4].

Finally, attempts to implement and use educational programming languages at schools were not successful. From our point of view, a way out is to find an industrial programming language, which can be successfully used as an educational one.

2 Requirements to Educational Programming Languages

Let's note that the Pascal language was initially developed by N. Wirth as an educational language. This language demonstrates such conceptual peculiarities, like strict typing and availability of means of structured (procedural) programming. And, by N. Wirth, this language should facilitate forming of a good style of algorithmic thinking, and, thus, programming. In particular, the author tried to make its syntax intuitively clear even at the first acquaintance with the discipline "Programming".

During long time Pascal was fairly considered one of the best programming languages for education purposes. Unfortunately, the versions of programming environments, which were used to teach the language (Turbo Pascal, Borland Pascal), have become outdated. New programming systems, which are based on Pascal, for example, Delphi, are too expensive and industrially, not educationally, oriented. Particularly, the programming environment is poorly suitable to teach basic programming and algorithmization due to its complexity.

At the same time, modern means of interface are so well developed, that their existence should not be ignored even at the elementary stages of learning programming. From our point of view, the use of console for the input-output is nowadays insufficient.

Let's take a look at the requirements, which are necessary to be put forth to the educational programming language and to the programming system, connected with it. By A.P. Ershov, the following requirements to the educational programming languages should be marked out [4]:

- The language and the programming system, connected with it, should represent all main programming concepts in their structure.
- The language should have clear logic and should be methodically conditioned, i.e. its structure should be built by certain methodical scheme, which allows sequential introducing of new concepts and forming of necessary skills.

- The structure of educational language should be methodically conjugated with the structure of modern standard high-level programming languages, i.e. learning this language should make further transition to learning other languages (for instance, within professional training) easier or not complicated.
Let's add some more clarity to the description of educational programming language.
- Programming language should correspond to the next paradigms of programming: imperative and structural (at basic levels), and object-oriented programming (later stages).
- Educational programming language should be learnt in the way to avoid “learning in advance”, i.e. using the references to the materials, which haven't been studied before.
- The syntax of programming language should be as simple and clear as possible, to make a program not only easily written, but also easily read and understood.
- The language should have sufficiently high-level and a special emphasis on algorithmization. Moreover, mechanisms of its interaction with a computer should be hidden as much as possible. Therefore, the use of extra-high level features, such as a concept of abstract data type and its explicit definition in the text of the program (for example, like class), is advisable.
- The problem of the use of explicit memory management needs individual consideration. It is very important for studying, especially when learning dynamic data structures.

3 Requirements to the Programming Systems

Let's consider requirements to the programming systems for educational purposes. From our point of view, the main requirements are the following:

3.1. Requirements to the environment:

- Cross-platform – it is advisable not to bind a student to a certain platform, because a user should choose the platform to work on his/her own.
- Open Source –programming system should be an open one, with open source codes.
- Free distribution –programming system should be free of charge.

3.2. Requirements to the subsystem of editing:

- Initial files should be stored in the file system;
- There should be a system of projects, which jointly stores related files and provides convenient access to the files within the project;
- Modern editing tools of initial texts, like autoformat, intellisense, pages numbering, etc. should be available;
- Modern debugging tools, like step-by-step running, variables watch, call stack should be available.

3.3. Requirements to the libraries of educational programming language.

- Educational programming language should contain the libraries, which provide the solutions for main tasks, like file system, sockets, Internet, localization, user interface, etc.

4 Industry Value of a Programming Language

Under the concept Industry Value we mean the ability of the language to be used and demanded in IT sphere. Let's consider such programming languages, like Pascal, C#, Java, Python, C++, ObjectPascal, PHP, Javascript from the point of view of their applicability in different areas.

In the capacity of evaluation criteria let's consider the possibilities of writing the applications in the next areas: system programming, Web-programming, Desktop-programming, Mobile-programming, Web-client development. See Table 1

Table 2. Applicability of programming languages.

	Pascal	C#	Java	Python	C++	Object Pascal	PHP	Javascript
System programming	0	0	0	0	2	0	0	0
Web – programming	0	2	2	2	1	1	2	1
Desktop – programming	1	2	2	2	2	2	1	0
Mobile – programming	0	2	2	2	2	0	0	0
Web – client development	0	2	2	2	0	0	0	2

Legend:

0 – the language is impossible to use for such purposes

1 –the language can be used for these purposes but there are better solutions

2 – the language can be used for these purposes and is recommended to

There is a need to make a conclusion that certain programming languages are aimed to solve bigger amount of tasks. Therefore, it's better to choose them for educational purposes.

4.1 Rating of Programming Languages

As programming languages constantly evolve, there are some systems of their evaluation.

Dutch company TIOBE Software BV [9] is a famous author of the popularity rating of programming languages, which is calculated on a regular basis. While making a popularity rating, Tiobe takes into account the amount of language experts, the amount of language learning courses as well as the amount of vendors, who support the language and the amount of the code, indexed by search engines. Let's take a look at the popularity rating of programming languages for February, 2012.

Table 2. TIOBE rating.

	Place in a rating
Pascal	15
C#	3
Java	1
Python	8
C++	4
Object Pascal	-
PHP	6
Javascript	10

Ohloh.net is a free, public directory of Free and Open Source Software and the contributors who create and maintain it. Ohloh represents a resembling rating, based on the activity of commits (saving) of program code.

The lines show the count of monthly commits made by source code developers. Commits including multiple languages are counted once for each language [10].

It is clear that the high rating of a programming language should be taken into consideration when choosing a programming language for educational purposes. Due to the ratings, represented at the table 2 and figure 1, it's possible to make a conclusion that any listed programming language is more popular and needed, than Pascal.

5 Choosing a Programming Language

Due to all the information represented above, let's try to choose the best programming language to replace Pascal.

As far as the only drawback of Pascal is reducing of its use in real projects, and, as a result, falling of its popularity, choosing a popular and demanded language makes sense. All programming languages, taken into the consideration, meet this condition.

The second important criterion is a level of suitability of each programming language for teaching, including beginner level.

Let's examine the chosen languages one by one.

Pascal – is a good language to teach programming, but the use of this language in real applications is on the wane, and newer languages are replacing it. Moreover, existent programming environments are either outdated or proprietary, or experimental. The second considerable reason to refuse this language is a poor motivation of students to learn it due to the practical impossibility to use it in real projects.

C#, Java – are similar languages with resembling concepts and tasks. They both are very important in the IT field. Unfortunately, they are difficult for beginners because require the knowledge of object-oriented programming almost immediately. The same

drawback is characteristic of C, Java, C# languages, which is why they are not suitable to teach beginners.

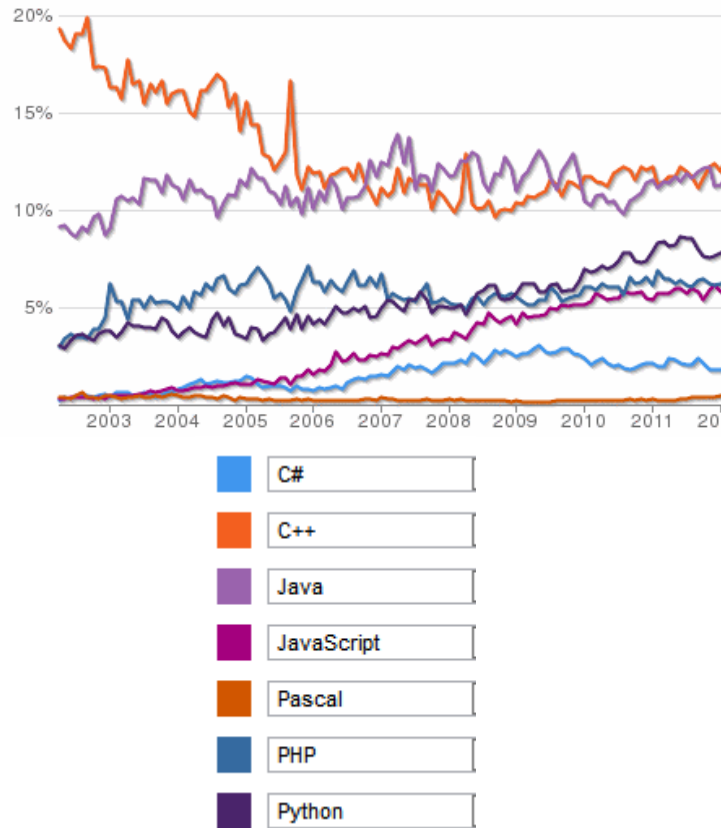


Fig. 2. Dynamics of the activity of the use of programming languages.

Example of the Java code (Hello world):

```
public class helloWorld
{
    public static void main(String [] args)
    {
        System.out.println("Hello World!");
    }
}
```

C++ - is now one of the basic and the most popular languages. It is very convenient to teach basic skills of system programming to programmers. It should be used to teach specialists, but not beginners [8].

ObjectPascal – is Pascal which supports the paradigm of object-oriented programming. It is used as a basic language of Delphi programming system. Because of the price of the programming system and the necessity to operate with the concepts of object-oriented programming, it is not suitable to teach programming to beginners.

PHP – is a programming language used to build server scenarios for web sites. Despite its popularity, PHP language is not very elegant and features syntax contradictions, non-coordination in function names and a dynamic typing of variables with a possibility to change a type. PHP possesses C-like syntax, and is practically used only in web programming. Therefore, PHP is not suitable to teach programming to beginners.

Javascript – is an object-oriented scripting language. It is a dialect of ECMAScript language. JavaScript is usually used as embedding language for program access to application objects. It is most widely used as a scenario language in browsers, to add interactivity to web pages. Its main architectural features are: dynamic typing, weak typing, automatic memory management, prototype programming, has first-class functions. This language is not suitable to teach programming to beginners because of the peculiarities of its use [12].

Python – is a high-level programming language for joint purposes, with an emphasis on a developer's productivity and readability of the code. Main architectural features are: dynamic typing, complete introspection, exceptions processing mechanism, multithreaded computations support and easy-to-use high-level data structures. It will be regarded as a pretender to replace Pascal language.

6 Python as a Programming Language for Beginners

Python is a modern scripting language, which supports main programming paradigms. [3, 5].

The philosophy of the language «The Zen of Python» requires elegant style of writing the programs from programmers:

‘Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.’ [11]

As a quality of introduction of the language into teaching process depends on teachers, one of the main tasks during the introduction will be teachers' retraining to teach the cycle of disciplines with the use of Python. Notably, there should be no problems with mastering the language for the beginner level of programming, because the syntax of Python is very simple. In addition, Python's syntax is not C-like, which makes understanding of programs and algorithms easier.

Let's compare main algorithmic constructions of Pascal and Python.

Example of the code in Pascal (Hello world):

```
Program test;
Begin
  Writeln('Hello world');
End.
```

Example of the code in Python (Hello world):

```
print "Hello world"
```

Example of the code in Pascal (branching):

```
Program test;
Var
  x,y,z:integer;
Begin
  X:=5;
  Y:=2;
  Z:=3;
  If x<y and y<z then
    Writeln('Good');
  Else
    Begin
      Z:=x+y;
      Writeln(z);
    End;
End.
```

Example of the code in Python (branching):

```
x,y,z = 5,2,3
# Another way
#x=5
#y=2
#z=3
if x < y < z:
    print "Good";
else:
    z = x+y
    print z
```

Example of the code in Pascal (cycles):

```
Program test;
Var
  x:integer;
Begin
  x:=5;
  While x<10 do
    begin
      Writeln(x);
      x:=x+1
    end
End.
```

Example of the code in Python (cycles):

```
x=5
while x < 10:
    print x
    x = x + 1
```

Example of the code in Pascal (functions):

```
Program QuickSort;
Const
  n = 5;
Var
  A : array[1..n] of integer;
Procedure Swap(i, j : Integer);
Var
  b : Data;
Begin
  b := a[i];
  a[i] := a[j];
  a[j] := b
End;

Procedure Hoare(L, R : Integer);
Var
  left, right : Integer;
  x : Data;
Begin
  If L < R
  then begin
    x := A[(L + R) div 2];
    left := L;
    right := R ;
    Repeat
      While A[left] < x do
        left := left + 1;
      While A[right] > x do
        right:=right - 1;
      If left <= right
      then begin
        Swap(left, right);
        left := left + 1;
        right := right - 1;
      end
    until left > right;
    Hoare(L, right);
    Hoare(left, R)
  end
End;

Begin
  A[1]:=2;
  A[2]:=7;
  A[3]:=1;
  A[4]:=5;
  A[5]:=2;
```

```

    Hoare(1, n);
End.
Example of the code in Python (functions):
def hoare(l, r, arr):
    if l<r:
        x = arr[int((l+r)/2)]
        left = l
        right = r
        while not(left>right):
            while arr[left] < x:
                left = left + 1
            while arr[right] > x:
                right = right - 1
            if left <= right:
                arr[left],arr[right] = arr[right],arr[left]
                left = left + 1
                right = right - 1
            hoare(l, right, arr)
            hoare(left, r, arr)
    return arr

vector = [2, 7, 1, 5, 2]

vector = hoare(0,len(vector)-1,vector)

print vector

```

Among the advantages of Python there is a need to note that the interpreter is a cross-platform one; there is a big amount of third-party and standard libraries, which enable solving of almost any task; the language is supported by the development environment Eclipse.

Python is ported and works properly on all famous platforms. It is possible to program for the .NET platform on Python.

Among drawbacks of the Python language for educational purposes, it is possible to name dynamic typing and impossibility to use references. If the possibility to use references refers to the part of system programming and was excluded intentionally, dynamic typing is undesirable for educational programming language. This is the main shortcoming of Python in the context of using it for teaching.

7 Conclusion

The use of Pascal language to teach programming to beginners at schools and in universities has become old-fashioned. The discussion around adequate replacement of this language has lasted for quite a long time. Now Python language, which is powerful and simple, is suggested to be used for these purposes.

Python supports lots of programming paradigms: structural, object-oriented, functional, imperative and aspect-oriented, and learning can be started without any

preparation. There is one more advantage of the language: all algorithms are written easily and structurally in Python. Therefore, due to all mentioned above, it is possible to affirm that Python pretends to become a decent replacement for educational programming language PASCAL both at schools and on the first courses of higher education establishments. Programmers with strong knowledge of this language are always in demand, and they're well paid. Hence, the students, who mastered Python, can aspire to start positions in IT companies.

References

1. Zelle, J. M.: Python as a First Language, <http://mcsp.wartburg.edu/zelle/python/python-first.html>
2. Dr. Donaldson, T.: Python as a First Programming Language for Everyone, <http://www.cs.ubc.ca/wccce/Program03/papers/Toby.html>
3. [http://en.wikipedia.org/wiki/Python_\(programming_language\)](http://en.wikipedia.org/wiki/Python_(programming_language))
4. <http://ershov.iis.nsk.su/archive/eaimage.asp?did=41919&fileid=224284>
5. <http://python.org/>
6. Spivakovsky, O.V., Lvov, M.S.: Ways of Improvement of the Course "Basic Algorithmization and Programming" at the teachers' training higher education establishments. In: Computer at School and in Family. vol.4, pp.22-24 (2001) (in Ukrainian)
7. Spivakovsky O.V.: The Concept of Information Science Teaching at School and in Teachers' Training Higher Education Establishment . In: Computer at School and in Family. vol3., pp. 21-25 (2003) (in Ukrainian)
8. Stolyarov A.V.: Essay "C Language and Teaching Beginning Programming", http://www.stolyarov.info/pvt/anti_c . (in Russian)
9. <http://www.tiobe.com>
10. <http://www.ohloh.net>
11. <http://www.python.org/dev/peps/pep-0020/>
12. <http://en.wikipedia.org/wiki/JavaScript>
13. Phillips D.: Python 3 Object Oriented Programming. ISBN-13: 978-1-849511-26-1 Packt Publishing, 404 pages (July 2010)
14. Alchin M.: Pro Python. ISBN-13: 978-1-4302-2757-1 Apress, 368 pages (June 2010)

Creation of Multimedia Guides to the History of Music as a Means to form Professional Competence of Future Music Teachers

Lyudmila Gavrilova¹

¹SSPU, Slavyansk State Pedagogical University
Street General Batyuk 19, 84112 Slavyansk, Ukraine
lusjamuz@mail.ru

Abstract. The article exposes an urgent problem of contemporary university education, specifically the one of designing of electronic manuals on artistic disciplines. It provides redefinitions of the terms nomenclature (“multimedia aids”, “electronic manual”) and the requirements for the modern multimedia aids of teaching. The authors present the electronic textbook “Russian Music: from Ancient Times to Early 20th Century”, define pedagogical objectives determining the necessity to introduce the textbook to the study of the history of music. Consequently, the structure of the manual and various possibilities of its use in teaching music students of both pedagogical and professional specialties are being analyzed.

Keywords: multimedia textbook, history of music, Russian music.

Key terms: teaching process, ICT component, development, integration, standardization process

1 Introduction

The structure of professional competence, essential for future music teachers, has recently supplemented with such a component as informational competence. In Ukrainian educational studies of the last decades the idea was asserted that teacher’s informational competence is a component of the teacher’s general pedagogic culture, being one of the most important indicators of teaching skills and their compliance with international standards in higher education [1]. Use of information and communication technologies (ICT) in teaching practice proved their advantages over traditional methods, since the former are more efficient and correspond better to the principles of individualization and differentiation, intensification and effectiveness. A modern music teacher has to acquire skills of ICT use, provide multimedia aids on classes, create his own multimedia issues, maintain software programs for music education.

It must be mentioned that nowadays there is already a great number of multimedia products of educational and informative nature, among which the most interesting are the following:

- multimedia encyclopedias of general cultural character (World Artistic Culture, a software study book, aimed at supporting the same academic course for the 10th and 11th forms at the schools of Russian Federation; numerous multimedia encyclopedias dedicated to the masterpieces in different kinds of art, produced by private company New Media Generation in association with Cyril and Methodius Company); Sonata: World Culture in the Mirror of Music, created by L. Zalesskiy and the company Three Sisters; Encyclopedia of Theater, worked out by the electronic publishing house Cordis & Media; Encyclopedia of Foreign Classic Art, designed by the companies Interactive world and KOMINFO, as well as many other resources);
- multimedia projects aimed at learning music in different aspects: Music Class, a music trainer for children, designed by New Media Generation Company; electronic Piano School, created by S. Pridvorov; Music Trainer, worked out by V. Belobrov and Adept Company and IDDK; Music Flash Composer, providing the possibilities to improvise with timbers of several instruments, etc.;
- numerous special programs for music ear training, designed by foreign authors, which propose exercises aimed at discrimination of music intervals, chords, tonalities, and sequences of chords, rhythmic exercises, etc.: Ear Power, produced by the American company Fast & Soft; the program Earope, designed by the Danish company Cope Media; the program Auralia, issued by the Australian company Rising Software, as well as other software programs;
- encyclopedias of various musical instruments, which can be used in teaching music for different age groups: Terra Musicalis: Virtual Museum of Musical Instruments, designed by Hyper Method, presenting a unique collection of Saint Petersburg Museum; Encyclopedia of Musical Instruments, produced by company KorAx, which demonstrates various instruments, their structure, peculiarities of their acoustics;
- educational program Musical Art for primary and secondary school, created on the basis of the music curriculum of comprehensive school, so every lesson presents the corresponding topic from the curriculum, provides pictures, texts, animation, video and audio clips, pieces of music, songs, and their karaoke versions.

However such a variety of multimedia production which is useful for learning music at comprehensive school is not efficient for the professional training of music teachers. At the same time we have to mention that certain experience has already been gained concerning creation of software study aids in certain technical disciplines, foreign languages, informatics, etc. Theoretical consideration was given to the requirements to electronic educational editions. O. Krasovsky [2], V. Madzigon, V. Lapynsky [3], and other Ukrainian authors worked out a set of principles that determine structuring of such texts, and analyzed pedagogical aspects of designing and implementation of the electronic teaching aids.

The aim of the current paper is to present the textbook on the history of music as well as to analyze methodological approaches to its use in training future music teachers. The authors of the textbook are L. Gavrilova, Ph.D., and V. Sergiyenko, Doctor of Education. The textbook on the history of musical art *Russian Music: from*

Ancient Times to the Early 20th Century was created as the basic studybook for the course *History of Russian Music*, obligatory for the students of Slavyansk state Pedagogical University (specialty 6.010102 – Primary Education, specialization: Music). The textbook can be used not only in the system of art education in pedagogical universities, but also in professional music education, since similar courses are offered in music schools and conservatories.



Fig. 1. The cover of the textbook.

The multimedia training product on the history of musical art *Russian Music: from Ancient Times to the Early 20th Century* is an electronic textbook which in contrast to the traditional printed editions contains visual and audio components. The textbook is meant to be used at university, so its structure and content follow the requirements for the design of electronic textbooks for high school [4, 5] and include:

- a management system, providing means of structuring the educational material, tests and feedback;
- methods that accelerate the learning process, such as hypertext and hypermedia;
- graphical tools that provide effective use of visual aids;
- test system that enables to control students' knowledge.

Textbook materials are presented on 2 discs: the first DVD-ROM contains the textbook itself and the second is a compilation of music for individual listening. We should also mention the hardware requirements needed to install the tutorial, which are as follows:

- Processor 1000Mhz (recommended 2000 Mhz);
- Memory: 512 Mb (1024 Mb);

- Hard Disk 3 Gb of free memory;
- Operating system: WindowsXP, Vista (32/64) Windows 7(32/64-bit).

It is important to consider the structure of the textbook and the possibility of its use in training future music teachers.

Installation of the textbook is traditional: one should insert the CD into the CD drive then select "autorun.exe" in the menu, which will lead to a startup tutorial.

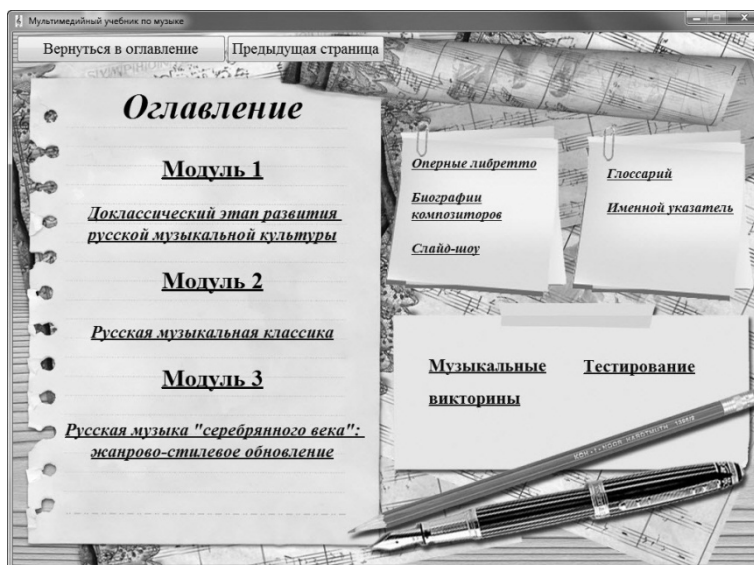


Fig. 2. Menu.

The course of the history of Russian music covers the period from ancient times to the early 20th century that is from the early Middle Ages to the so called Silver Age of music in Russian culture. The textbook is opening with the Menu and it is structured as follows:

- lecture (a text with hyperlinks to the glossary, index of names, biographies, opera librettos);
- multimedia slide show;
- glossary (dictionary of musical terms);
- index of names;
- opera librettos;
- biographies of composers;
- musical quiz in two variants that is aimed at controlling students' knowledge of certain compositions;
- tests in two variants.

Music for individual listening is presented on a separate mp.3 disk. Students are offered to listen to the following works by Russian composers:

- a sufficiently large collection of Russian sacred music (from ancient chants to the works by anonymous composers of Synodal School of the late 19th century, and the finest examples of sacred music of Sergei Rachmaninoff);

- masterpieces of Russian chamber and vocal music;
- works of Russian musical theater: the most famous operas and ballets that are traditionally studied in the course of the history of Russian music (Glinka's *Ruslan and Lyudmila*, Ivan Susanin, Dargomyzhsky's *Rusalka*, Mussorgsky's *Boris Godunov*, Borodin's *Prince Igor*, Tchaikovsky's *Eugene Onegin*, *Queen of Spades*, Stravinsky's *Petrushka* and *The Rite of Spring*);
- the best examples of instrumental music of Russian composers (symphonic works by Glinka, Borodin, and Scriabin, Tchaikovsky's symphonies, Rachmaninoff's piano concertos, separate instrumental miniatures and cycles).

The Menu opens navigation to any section of the textbook. Lectures are set in the textbook according to the historical principle. Culturological material is divided into 3 modules and 7 units.

Module 1. Pre-classical stage of development of Russian music: from ancient times to the late 18th century).

Module 2. Russian classical music (from Glinka to Tchaikovsky).

Module 3. Russian music of *Silver Age*: genre and style transformations (Rachmaninoff, Stravinsky, Scriabin, and others).

Texts of lectures, modules and clusters were composed with the reference to modern printed books on the history of Russian music (I. Rapatsky *History of Russian Music: From Ancient Rus' to the Silver Age*, Moscow (2001) (In Russian). In addition we used other books (E. Orlova *Lectures on the History of Russian Music*, Moscow (1979) (In Russian); B. Asafiev *Russian Music of 19th and 20th Centuries*, Leningrad (1978) (In Russian); T. Levaya *Russian Musical Culture of the Early 20th Century in the Context of Artistic Trends of the Era*, Moscow (1991) (In Russian)), as well as multi-volume history of Russian Music edited by A. Kandinsky, and Y. Keldysh, Internet sources (such sites as mus-info.ru, classic-music.ru, belcanto.ru, etc.). Glossary that includes 262 items and Index of names including 335 items have interesting selection of concepts from theory of music, music history and general cultural context. Informative sections of the book are supplemented with the biographies of 25 Russian composers, and 11 of the most famous opera librettos.



Fig. 3. Pages from Glossary (Spiritual Verse), and Index (Anna Akhmatova).

Slide show serves as a multimedia application to every topic, including widely available photographic materials, reproductions of paintings, audio fragments of music compositions and video clips of operas, ballets, concerts.



Fig. 4. Pages of slide show.

Special attention is given to the system of means aimed at controlling students' knowledge of theory and musical compositions. Traditionally while studying the course of history of music, students are supposed to penetrate the world of music through listening to numerous pieces of music. The multimedia textbook provides students with an opportunity to listen to the set of compositions, and use quizzes to test their knowledge:

- to identify certain audio fragment, to name the author, the composition and the part the fragment was taken from (an act in opera, a part of symphony);
- to choose the right answer among the given variants.

Passing each stage of the quiz, students can examine the record of their answers, compare them with the right options and find out the level of their knowledge. In addition, the textbook contains an actual test at the end of each unit, students are offered two variants of closed test, so they have to choose one or more correct answers.

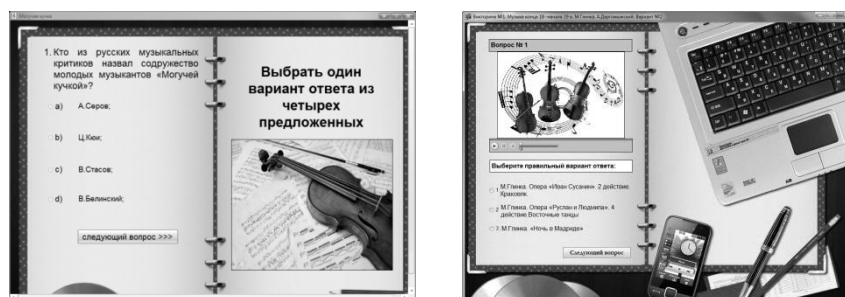


Fig. 5. Musical Quiz and Test.

The introduction of electronic textbook to the process of studying the history of music contributes to the solution of the following educational objectives:

- to study Russian musical culture as a whole, which absorbed spiritual experience of the nation and reflected religious, philosophical, aesthetic and moral principles of certain historical periods;
- to learn the main styles and trends in the Russian music of the period from ancient times to Silver Age, to get knowledge of the origins and the main stages of the evolution of Russian culture;

- to specify the features of different musical styles and trends in Russian music on the basis of certain compositions by the great Russian musicians;
- to develop feelings, emotions, imagination, and artistic abilities of students;
- to refine their artistic and aesthetic tastes, to bring up the need in personal assessment of values;
- to enlarge the scope of their thinking, to make them create the cultural space of their own.

The multimedia textbook on the history of music *History of Russian Music: from Ancient Times to the Early 20th Century* can be used in various spheres of higher education.

First, it can be used for lectures on the history of Russian music, when the lecturer's narration is followed by presentations of audio and video clips, photographs, reproductions of paintings, etc. Multimedia slide show can accompany the given facts of the composer's life and work, it can illustrate information about the features of the musical culture of certain age, exemplify specific traits of certain musical trend, etc. The lecturer has an opportunity to select the necessary information in accordance with the audience comprehension. The textbook can be used to introduce the new material as well as to fix and repeat the items already known to the students, it can be useful on practical classes, and at the stage of assessment of students' knowledge.

Students can work with the textbook in the classroom, the computers being provided, and independently as well. The textbook can be used:

- as a source of the following information: lectures, slides, audio and video clips, glossary, index of names, biographies of Russian composers;
- as a visual aid: the variety of multimedia components helps to get acquainted with the numerous masterpieces of art, sculpture, architecture, as well as to listen to the episodes of musical compositions, to watch the fragments of operas and ballets;
- to exercise in self-listening, and preparation to the quiz;
- to write papers and reports;
- to get ready for the practical classes;
- to reduce the gaps of knowledge in the history of Russian music;
- to estimate the level of knowledge.

The textbook can be used in a classroom with one computer and a projector, as well as in the computer lab.

Use of multimedia provides additional possibilities of presenting information on the history of music:

- the possibility to see the visual details of paintings, historical documents, old editions of music, etc.;
- the possibility to use hypertext and hypermedia links that simplifies the coordination of student's independent work;
- free navigation through the content, the direct access to the menu of the textbook.

The use of the multimedia textbook for the course History of Russian Music in Slavyansk State Pedagogical University allows to summarize the following:

- the use of multimedia aids based on simultaneous visual and auditory perception of the material, greatly increases the interest of students to the subject and study in general;

- students' independent work with the multimedia electronic textbook contributes to the development of cognitive abilities, stimulates the desire for self-improvement, awakens their imagination and creative thinking, enriches their professional potential;
- the use of ICT in the process of musical education determine the systemic study of phenomena of art and reality, sustains the correlation between different sciences and the sphere of art and culture.

All of this form the firm basis for development of professional competence of future music teachers and determine the way of becoming of a future musician, ready to face the challenges of the contemporary civilization.

References

1. Kolomiyets, A.: Informational culture of Primary School Teacher: Monograph. Vinnitsa (2007) (in Ukrainian)
2. Krasovsky, O.: The Main Structural Requirements for Electronic Textbooks for Secondary 12-Years School, <http://www.nbu.gov/> (in Ukrainian)
3. Madzigon, V., Lapynsky, V., Doroshenko Y.: Pedagogical Aspects of Creation and Use of Electronic Teaching Aids. In: The Problems of the Modern Study book, pp. 70--78. Kyiv (2003) (in Ukrainian)
4. Kademina, M., Shestopalyuk, O.: Electronic Textbook on Interaction Basis, <http://www.nbu.gov/> (in Ukrainian)
5. Volynsky, V., Krasovsky, O., Chornous, O., Yakushina, T.: Structure and Content pf Electronic Textbooks: Cognitive and Behavioral Aspects. In: Computer at School and Family, pp. 44--49 (2011) (in Ukrainian)

An Experience of the Creation and Approbation of the Learning Course “NIT and TFE” for Future Teachers

Nataliya Kushnir¹ and Anna Manzhula¹

¹ Kherson State University, 40 roktiv Zhovtnya St., 27, 73000 Kherson, Ukraine
kushnir@ksu.ks.ua, ilovetrees@mail.ru

Abstract. The article presents the experience of forming the ICT-competencies of future teachers of elementary school and early education. The course is developed on the Moodle platform for distance education. The course objective is training of future teachers to use ICT in their pedagogical activities, understanding the role of ICT, its opportunities and perspectives for improving educational process.

Keywords. Distance learning, Moodle, ICT-competencies, future teachers of elementary school.

Key Terms. Teaching Process, Teaching Pattern, ICT Tool, Competence Formation Process.

1 Introduction

The tendency of the development of information society, changing the teacher's role from a source of knowledge to facilitator and the approval new state standard of elementary school, which includes Informatics as required course for all pupils starting from the second year led to increasing the priority of the course “New information technologies and technical facilities of education” (NIT and TFE) for professional training of future elementary school teachers.

Now lecturers of this course observe such tendency: students use the ICT more often and have better skills of working with different applications. The data, got from the entrance survey, have confirmed it. Really, there is no sense to repeat the school course of Informatics: 89% of students have their own computer or laptop; 70% of respondents have free access to the Internet outside university; 78% of students have stated they are skilled in using Office programs, Internet, printers, scanners; 51% of respondents have experience of self-reliant learning of software. 49% of students feel comfort and enjoy working at the computer, 33% – feel assurance. So, there is a need to move accent from the learning “buttons” in Informatics to understanding ICT creative potential for teaching and applying it in future pedagogical activities.

It had identified a need to research changes of goals and assignment, content, methods and forms of teaching this course, peculiarities of its realization and analysis of received results.

2 The Essence of the Research

The course “NIT and TFE” is a single discipline in computer sciences in the curriculum for undergraduate students of specialty “Primary education”, except students studying at specialization “Basics of Informatics”. The course “NIT and TFE” is a base for studying the discipline “New information technologies in preschool education” (8 hours of lectures and 12 hours of practical lessons) for undergraduate students of the specialty “Preschool education”. The overall objective of these disciplines is training of future teachers to use ICT in their pedagogical activities, understanding the role of ICT, its opportunities and perspectives for improving educational process.

The objective of the course implemented such educational and training tasks:

- to uncover the importance of role, opportunities and perspectives of the ICT in preschool and elementary education, train the skills of studying with the ICT and reasonable using these technologies in future professional activities;
- to familiarize the students with the basic facilities and methods of the modern information technologies, their theoretical and technical basis;
- to give to the students knowledge, abilities and skills, which let them learn and develop independently having in mind the effectiveness of using ICT in their future professional work;
- to organize students’ creative activities in making their own computer programs for teaching;
- to give an opportunity to each student to realize his educational trajectory in a way of differentiation and increasing the quantity of creative tasks;
- to encourage students to make a collection of e-resources for learning;
- to form the basics of information culture of the students.

The importance of the educational and training tasks listed above requires a priority status of this and similar discipline in the curriculum of pedagogical specialties. In our opinion, it would be reasonable to extend this course and add a lot of academic hours at the expense of variation part of the curriculum. For example, at the Department of Preschool and Elementary Education of Kherson State University (DPEE and ES KSU) there is only 18 academic hours for work in auditorium: 4 hours for lectures and 14 hours for practical lessons, while the quantity of time for independent work is 36 hours. It requires making clear tasks, forms and terms of the students’ work. It was confirmed by the results of final poll in which fair quantity of students had expressed an opinion to add a lot of academic hours and extend this course.

The actual level of the development of Internet-services lets an educator write an electronic summary (e.g. in form of presentation), regularly change content without any expenses for re-edition. Students can use such electronic resources in any time, if they pass a lesson or wish to revise the material. Students can get distance

recommendations from the educator or other students about different questions concerning their tasks. Students can also load the files with an executed work to the site for control. Students have the opportunity to evaluate the course and a quality of its teaching and in this way the administration can perform monitoring. This type of organization of studying has already become a standard in many leading universities all over the world.

The course described above satisfies all these norms and the Moodle platform for distant learning helps even the beginner to realize it conveniently and easy (see <http://ksuonline.ksu.ks.ua>).

Let us describe the course in detail.

The main webpage visualizes a structure of the course; it consists of the annotation to it and the blocks with hyperlinks to the each topic. The course contains an entrance poll and testing, they help to learn about the audience better, identify the level of residual knowledge of Informatics of the educators that gives an opportunity to optimize future educational process. According to the results of the entrance test students are subdivided in two levels of difficulty of the tasks: intermediate Level A and advanced Level B. The course includes also a final poll. Despite the fact the students spend a lot of time at the computer, they use it mainly for entertainments. So the level of residual knowledge of the school Informatics course is low and 79% of students studied at level A, respectively 21% performed the tasks of level B. The latter level we proposed to the students, which showed high results of entrance test: a quantity of right answers had to amount more than 75 %.

The content of lectures includes the basic definitions of ICT, approaches to using ICT in education and classification of pedagogical software, new tendencies as Web 2.0 and so on. Theoretical part of the course also contains modern requirements of elementary teacher's ICT-competency. The priority of the course is its professional focus, so content of lectures includes the important questions such as peculiarities and approaches to using ICT, sanitary and hygienic norms of working at the computer in kindergartens and elementary schools, recommendations for choosing pedagogical software.

We formulated the following requirements to the practical tasks for students on the ground of methodological literature analysis:

1. Using ICT in the educational process, on the one hand, facilitates to the systematization of student's knowledge in this sphere, on the other hand, reduces the level of creative activities. Therefore, the need to increase amount of creative tasks appears to be the compensation for such ICT influence.
2. Tasks should be directed to the forming self-education skills that is one of the factors of the further professional development.
3. Tasks should raise internal motivation for education, particularly for studying opportunities of using ICT in the future teaching practice.
4. Tasks should be oriented to the future professional activities.
5. Tasks should form understanding of modern tendencies of the ICT development and using it in the educational process.

Practical tasks are structured in four blocks: "Information and communications", "Creating and using an educational presentation", "Text documents for a teacher", "Using of Excel program in pedagogical practice" and distinguished between two levels: intermediate Level A and advanced Level B. Students are subdivided on two

groups according to the results of the entrance test (see Table 1). We consider starting of the course with the topic “Information and communications” to be principally important, because not every student has an experience of distance learning and unfortunately a lot of students even don’t have e-mail or don’t use it. Therefore, we had to teach students to use the Moodle platform for distant learning and give tutor’s consultations in a case of difficulties or technical problems with electronic correspondence.

Table 1. The plan of the course “NIT and TFE”.

The academic hours	Theme blocks	Auditorium tasks			Independent work	
		The content of the tasks		The forms of control	Content of the tasks	The forms of control
		A (intermediate)	B (advanced)			
2/1	Information and communications	Search of the information on the theme: “Educational programs for children”, adding the web-pages to the “Favourite”, filling the table following the example for 5 sites.	Creating the site by the tools of Google sites, filling the main page and the page of the useful recourses (minimum 5).	Loading the file to the site. The hyperlink to the blog on the educator’s e-mail box.	The acquaintance with the rules of network etiquette. Formulating 5-10 rules of using the Internet for children of primary school age.	The answers in a text form.
2/2		Forums and polls. Searching the forums of the professional themes, registration and participation in one of them. Creating a poll.	Creating the poll on the free service and allocation it on the own site.	The hyperlink to the site on the educator’s e-mail box.	The acquaintance with the document “ICT competency standards for teachers”.	Test on KSU Online. Loading the file to the site.
2/5	Creating and using an educational presentation	Producing “diapositive” tale with music support	Creating an interactive visual aid.	Loading the file to the site.	The acquaintance with the document “Design of educational products in MS PowerPoint”. Making test with triggers.	Loading the file to the site.
2/4		Creating didactical game with the hyperlinks.	Creating didactical game with the triggers.	Loading the file to the site.	Making the design of all presentations.	Loading the file to the site.

The academic hours	Theme blocks	Auditorium tasks			Independent work	
		The content of the tasks		The forms of control	Content of the tasks	The forms of control
		A (intermediate)	B (advanced)			
2/2	Text documents for a teacher	Making the letter of commendation by merging the documents.	Creating labels for a student's exercise book.	Loading the file to the site.	The "It would be interesting to know"	Loading the file to the site.
2/1	Excel program in pedagogical	Statistical data analysis using MS Excel.	Statistical data using MS Excel.	Loading the file to the site.	Drawing the graphics and diagrams.	Loading the file to the site.
2	Final lesson	"Children and the computer": search of information, statistical data, its graphic presentation in MS PowerPoint.		Loading the file to the site.		Final complex test. Test. Conclusive poll.

Among practical tasks there are such ones as creating and editing a site by the tools of Google Sites, filling its informational content; communicating in a blog or forum on the professional topics; producing "diapositive" tale with music support; creating interactive visual aids and didactical games by means of MS PowerPoint; creating labels for student's exercise book; making letter of commendation by merging documents in MS Word; statistical data analysis using MS Excel etc.

We've got interesting results of anonymous final students' poll. They noticed that the most difficult creative tasks are the most attractive and useful for them. The data showed 64 % of respondents answering the question "What task was the most difficult one?" chose following answers: "Creating a didactical game with the hyperlinks", "Making test with triggers", "Producing "diapositive" tale". As an answer to the question "What task do you consider to be the most useful for you?" 79% of students have selected variants "Producing "diapositive" tale", "Creating a didactical game with hyperlinks", "Making letter of commendation by merging the documents". About 67% of students gave the following answers: "Producing "diapositive" tale", "Creating a didactical game with hyperlinks", "Making the poster "It would be interesting to know", "Make letter of commendation by merging the documents" on the question "What of the tasks did arouse the greatest interest?"

We have to mention that statistic data presented earlier characterize answers of students, who studied at Level A (79% of the whole group). Approximately such statistical data describe poll's results in the group of Level B.

The practical tasks have typical structure: a title, an objective, necessary software, criterion of assessment, examples of executed task, step-by-step instruction with illustrations. Also we have submitted video-lessons and put hyperlinks to subsidiary web-resources as collateral relief. Observing the network etiquette was the important requirement to the students' work.

Everyone can review the best students' works on the site "Virtual gallery of creative multimedia works of DPEE KSU students"

<https://sites.google.com/site/museumfdpo>, which shows the real example of using free internet services in educational aims.

Among the most original products in gallery we wish to grant next works of our students: “In the vegetable garden” (Irina Govrishchenko), “The games” (Tetyana Kydrevs’ka) and “Compare numbers!” (Svitlana Shrub).

This site motivates students to the higher quality of creating multimedia educational products and cultivates respective attitude to the copyright. Furthermore, the students’ participation in creating of the open collection of multimedia projects promotes further collaboration, exchange of experience among other teachers in future.

Tasks for independent work are not differentiated and do not require a high level skills of ICT using, directed to the generalization and deepening Informatics knowledge, understanding of the role and opportunity of using ICT in future professional activities. Students have the considerable quantity of time (36 hour) for the independent work that predetermined its clear organization (statement or problem, forms of control, execution terms etc.). Moodle Platform provides a possibility for an educator to regulate the time for execution of the tasks (for example, an access to the function “loading files” will be closed for students after the deadline).

It is well known that principle “effort here and now, but result – later” reduces the motivation. Therefore it’s important that all tasks should be checked and marked before the next lesson and if students get some educator’s comments or recommendations, they can improve their product and, naturally, get a better mark. Students submit a result of their work in form determined previously by educator like “submit file on the site”, “send hyperlink to tutor’s e-mail”, “test on the site” and etc.

The authors added a section “Recourses for You” as extra support for students, which contain the presentations of lectures, “Useful resources” – structured collating of Internet-resources, “Main definitions” (glossary). ‘Guide of algorithms’ contains instructions illustrating some operation of using MS Office. Such reference books decrease the task description, because elements of algorithms are not duplicated. The structure of such tasks is convenient both for “weak” and for “experienced” software users: it gives step-by-step illustrated instructions for the first ones, and it does not draw the advanced users’ attention to unnecessary detailed description that let them move quickly and independently.

The final block of the web-page helps an educator to organize the last lesson that includes complex task (55 minutes) and final test (20 minutes). Students must show skills of using as a minimum three of four program tools considered on the practical lessons and abilities to search and analyze information, identify a problem question, well-reasoned presentation of certain point of view and graphic visualization of data, particularly statistical. So we decided to choose sufficiently broad a general theme of the final task – “Children and computers”.

The aim of the final test is to check the main learnt statements of the lecture material. Test includes questions of different form with different quantities of the right answers. Every student has only one attempt to take the test.

The aim of the conclusive poll is to detect the level of student’s satisfaction by quality of methodical support of the course, to reveal the most interesting and useful topics for them, to identify new strategies of improving the course.

Thereby, developed course “NIT and TFE” fully corresponds to the actual level of educational process organization at the universities. The course can be “disseminated” at other educational institutions without attracting the authors of this course to teaching and organization lessons. At the same time, the electronic form of this course is open for modification and improvement.

3 Results

According to the schedule of KSU working in 2011–2012 academic year this discipline had been teaching in period from the 1st of September to the 29th of December for third year students of the Department of Preschool and Elementary Education. Generally, during the period 109 students finished this course.

The results of final poll are evidence of students’ understanding the role of ICT in future professional activities. So, on the question “Do you consider the skills of working at the computer to be an integral part of your teacher’s activity?” 47% students gave an answer “Yes” and 42% – “Likely yes than no”.

Students have estimated the quality of the course materials “NIT and TFE” as 9.29 (arithmetical mean) from 10 and novelty of lectures’ information for them – 7.59 respectively, practical tasks – 7.64 according to the ten-point system.

The students described their emotional state during studying course “NIT and TFE” like the following: 42% of them felt interest, 17% –enthusiasm, 12% – astonishment, 8% –pride, 8% –joy, 6% –anxiety, 4% –terribly and 2% –happiness.

Consequently, the approbation has confirmed positive results of our work.

As for further improvement of substantive component of the course we can also follow the student’s wishes expressed in conclusive poll. Summarizing students’ replies the future course may include studying of other different software (for example, MS Publisher, MS Project, Adobe Photoshop, MS Access etc.) and studying programs as MS PowerPoint (triggers), MS Word, MS Excel, using Internet, creating sites etc. more detailed for using them at school. Also students mentioned a wish to study methods of ICT using in professional activities and a need to add the hours of auditorium lessons.

4 Conclusions

Detailed and clear structure of the course is useful for supporting equal conditions and criterions (ruling “double standards” out) and makes the process of estimation of students’ academic achievements independent from the educator. We want to note, that such organization of pedagogical process “disciplines” both students and tutors: requires their regular attention and higher level of activity. Students note when they had problems, they got help from a lecturer during the lesson – 54% and during the individual consultations – 9% (10% of students didn’t have any difficulties at all and 25% got a help from their colleagues).

The participation of students in this project has become the factor of enriching their experience of effective ICT using in education. Such system makes possible for a

student to act as a subject of the educational process: it also influences on the further improving the course and through the poll takes part in estimation of developers' and educators' work.

Moodle distance learning platform makes it possible to present the course visualized and structured. It makes the educators' work highly organized and "transparent". The potential of Moodle led to the general positive student's attitude to the course, particularly, because students can make some tasks distantly. The experience of development of this course by means of Moodle is useful for educators of any other discipline as well.

One of the aims of the article is to present the experience of forming the ICT-competencies of future teachers of elementary school and early education. So, we hope to get feedback relative to advantages and short comings of our approach.

References

1. Anderson, J., van Weert, T.: Information and Communication Technology in Education: A Curriculum for Schools and Program of Teacher Development. UNESCO, Paris (2002), <http://unesdoc.unesco.org/images/0012/001295/129538e.pdf>
2. ICT COMPETENCY STANDARDS FOR TEACHERS, Paris (2007), www.unesco.org/en/competency-standards-teachers
3. The course "New information technologies and technical facilities of education", <http://ksuonline.ksu.ks.ua/course/view.php?id=10>
4. The site "Virtual gallery of creative multimedia works of students DPEE KSU", <https://sites.google.com/site/museumfdpo/%20>
5. Kushnir N.O., Manzhula A.M.: The practical tasks of the course "NIT and TFE" for students of pedagogical specialties. Scientific pedagogical journal "Parus", Mykolayiv (2012) (In Russian)

Scientific and Educational Project “IT-OSVITA” as a Part of the Training System of Specialists for the Needs of IT Industry of Ukraine

Ganna Lomakovska¹, Nadiya Omelchenko², and Galyna Protsenko³

¹ Lyceum of Information Technologies №79, Kyiv, Ukraine
alomakovska@gmail.com

² “IT-Osvita” program

Nadya.Omelchenko@octava.ua

³ Education department, Incom

galina.protsenko@incom.ua

Abstract. The current development stage of the Information Society is constructing a new mission for education and a significant increase in the requirements for the training of specialists who are capable of self-realization and continued education throughout their life. The objectives of the education systems are to improve specialist training in information and communication technologies (ICT), as well as widespread implementation of ICT in teaching and learning processes, increasing the attractiveness of education and strengthening relations with the professional world. The task of educating personnel who are able to develop new information technologies (IT) and use them effectively in practice becomes strategically important. It is necessary to develop a national system of education in the IT sector, which will be in demand by the science and practice, to solve the problem. This article describes the conceptual approaches to forming an effective mechanism of interaction between secondary and higher education institutions and the main priorities of IT companies in Ukraine, which are based on the experience and recommendations of international and domestic organizations that specialize in modeling the organization of the education process in educational institutions specialized in preparing professionals for ensuring the needs of the IT-Industry in Ukraine.

Keywords. IT specialists, the problems of training, the mechanism of interaction between secondary and higher education institutions and IT companies in Ukraine.

Key Terms. Cooperation, Knowledge Transfer, Development, Deployment, Management, Industry

1 Introduction

The movement of humanity to the Information Society is gaining an increasingly rapid and revolutionary character. Especially rapid and noticeable changes are in the field of ICT. This fact leads to the state where the existing forms and methods of training in this area do not correspond with modern requirements and need a fundamental transformation. The key principles and strategic direction of this transformation should be the maximal approach of the education process to the requirements of the IT industry [5].

Training for the IT sector In Ukraine is realized by more than 250 higher education institutions of various forms of ownership and levels of accreditation. Before 2010, training for IT fields was carried out in 19 specialties and the degree levels offered were bachelor, specialist and master in accordance with the list of directions and professions training, approved by the Cabinet of Ministers of Ukraine May 24, 1997 № 507 "About the list of directions and professions, which are exercised in universities on the appropriate education and skill levels."

The changes that have occurred in the IT sphere, the needs of the labor market and the propositions of higher education institutions were taken into account during the forming of a new list of directions entitled "The List of Specialties" which was approved by a resolution in August 27, 2010 № 787 "About the approving of the list of specialties, which are training specialists in higher education institutions for education and skill levels of specialist and master").

The contents of IT sector training in Ukraine is harmonized with international recommendations of Computing Curricula adopted by the European and American scientific communities for the quality of training of IT professionals. Industry standards of higher education, which were developed and approved during 2009-2011, also generally correspond to international recommendations and educational programs of leading universities.

Institutions of higher education turned out 24465 personnel with higher education in the IT sector in 2011.

However, today domestic IT companies are experiencing an acute shortage of qualified IT professionals, especially in the public sector. According to the survey of the scarcest professions in Ukraine in January 2012, which was conducted by the International Personnel portal hh.ua, 6 out of ten jobs are programmers.

Therefore, the roundtables on the topic "The prospects for the IT industry in Ukraine" (speakers: Head of State agencies on Science, Innovation and Information V. Semynozhenko, the Chairman of supervisory board Octava Capital A. Kardakov, vice-president of the Association "IT Ukraine" I. Lisitsky, GD of "Microsoft Ukraine" A. Shymkiv) and "IT Education in Ukraine" (speakers: Head of State agencies on Science, innovation and Information V. Semynozhenko, Director of Business Development Ltd. "Infopulse Ukraine" O. Nehoda, the Head of administration of educational work of V. Dale East Ukraine University T. Morozova, Vice President of "Lyuksoft" D. Kushnir, director of Kiev Lyceum of information technology number 79 G. Lomakovska) were held on the initiative of the leading IT companies of Ukraine in 2010-2011.

The question about the preparation of highly professional staff who are able to develop new IT and effectively use it in practice was considered at a meeting of the

Government on September 21, 2011, in which Resolution "On approval of a plan to ensure the development of education in information technology in 2013" was adopted.

The working meeting of The Prime Minister of Ukraine, M. Azarov, with heads of Ukrainian educational institutions and representatives of associations of industry and leading IT companies in Ukraine, took place in February 2012.

In the network of initiatives and activities cited above, questions were considered concerning the level of student preparation in physics and mathematics, the development of programs and courses in information and communication technologies which are studied according to the choices of secondary school students, the establishment of profiling schools in leading universities, development of an effective mechanism of interaction between secondary and higher education institutions and IT companies in Ukraine in terms of training and the linking of market needs with education.

The development of secondary and higher education in Ukraine in order to ensure the IT industry has enough highly qualified personnel requires the development of an efficient mechanism of interaction between secondary and higher education institutions and IT companies in Ukraine. Close cooperation between secondary and higher education institutions, business structures and ensuring feedback from IT businesses become of particularly great importance in this context.

The purpose of this article is to describe the strategies and programs for implementation of effective models of the education process in general education that meets the modern social order concerning the training of specialists for the needs of the IT industry in Ukraine.

2 Review of the Science and Pedagogy Experiment "IT - Education"

The examination of the training process of future professionals shows the social order is considered to be a set of demands that society makes for education in the area of intellectual work and information technologies, among which the demands with particular importance are [6]:

- Ability to be inventive and think critically;
- Universal system knowledge, high adaptability and self-development;
- Key competence in the field of ICT;
- The ability to make decisions, and social responsibility;
- The ability to manage dynamic processes and to work with the project;
- Experience in team- work, high productivity.

The scientific and pedagogical experiment "IT - Education" was launched by the initiative of Taras Shevchenko National University of Kyiv together with the Association "IT Ukraine" and Incom Company on the base of the Lyceum of Informational Technologies № 79 in Kyiv and it provides for the beginning of junior programmer training at high school.

The main methodological features of the project were defined in the network of project "IT – Education". It found that it's main goal is to develop a new model of the educational process of training to meet the needs of the IT industry in Ukraine. The

basis for achieving the above objective is the interaction between institutions of secondary and higher education and employers of the IT - industry.

The concept of the science and pedagogy project "IT-education" was developed to achieve the above mentioned objective. In the content of the concept which is based on analysis of current training for ensuring the needs of the IT industry in Ukraine, substantive features of the problems arise and require immediate resolution. It was determined that achieving the defined goals of the project would require that the basic, strategic and priority objectives should include: increasing the quality of education in the context of psychological and vocational guidance of youth starting from secondary school and preparing students to be motivated in their selection of and training for mastering the specialty in the IT field during the learning process at the institution of higher education.

According to the order of the Ministry of Education, Youth and Sports of Ukraine on the terms of the experimental implementation of the Project, the following main stages of its implementation were defined. Phase I - June - August 2011 – the development of normative and legal support and the diagnostic study of the Project. Stage II - September 2011 - December 2012 – The development of education and logistics. III stage - installation of the training subjects (students, teachers, professors) at secondary schools and higher education institutions to perform tasks of psycho-educational training of IT experts (specialists). But, taking into consideration the fact that the task of each phase (as a result of the impact on the progress of the Project) is interconnected with each other, their execution was carried out comprehensively during the school year, and generally aimed at addressing the following main tasks:

- To develop organizational support for the project;
- To develop teaching resources of the project
- To develop a diagnostic system of ensuring of the project;
- To develop a technical and material method to ensure implementation of the training of subjects (students, teachers, professors) to perform tasks of technological, educational and vocational training (learning) for the future IT experts in higher education.

The evaluation of results of the II phase of the experiment was conducted in the form of vocational testing through the method of Madzhelano University. This has provided the ability to obtain statistical data which objectively characterizes the effectiveness of the educational process and dynamics of the intellectual activity of students of the experimental class of Lyceum of Informational Technologies № 79 in Kyiv. Professional orientation test has shown that the ability of students meets the professional interests and children have chosen the right direction in their future profession (fig. 1, 2).

According to the semantic features of the above-stated objectives and purpose of the project the following principles of its implementation were developed:

- The determination of normative conceptual state documents on education and the needs of the IT network for specialists of this profile;
- The Identification of stakeholders on the objectives of the project;
- The definition of criteria of evaluation and the summarizing of the results of the project's realization;
- The definition of the common business activities of the project;
- The determination of the management process of the project;

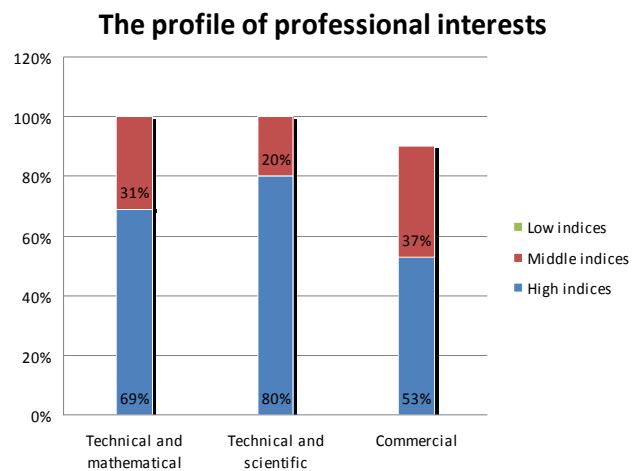


Fig. 10. The results of the test on the profile of professional interests.

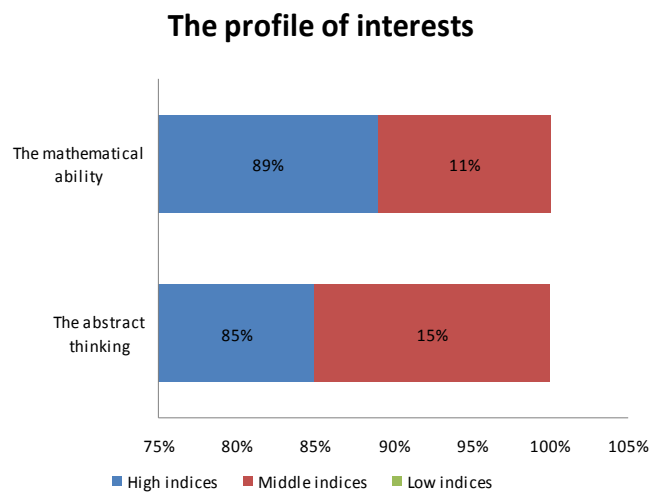


Fig. 11. The results of the test on the profile of interests.

- The definition of the training system as a subject of the project’s realization;
 - The definition of quantification and semantic features of the results of the project.
- According to the program of the science and education project "IT-education" for the 2011-2012 school year, in the context of activities and tasks a normative and legal standard was developed ensuring:
- Agreements with project participants were prepared and signed;

- A draft order was prepared that was aimed at implementing a research and educational project which is instilled into the form of the order of Ministry of Education, Youth and Sports of Ukraine 20.02.2011r. № 831 "On introduction of scientific-pedagogical project" IT Education";
- A concept of the science and pedagogy project was approved by the above order of 20.02.2011r. № 831;
- The composition of the science and education project coordinating council was created and approved;
- The site of the project (it-osvita.com.ua) was created;
- The discussion of seminar topics for the specialized training of teachers LIT number 79 KNU and teachers who are involved in the experiment was realized.
- Training and educational support for the objectives of IT Education was developed, especially:
 - The programs of special courses "Algorithms and programming", "Fundamentals of Project Management", "Introduction to the profession", "IT Ukraine", "English for IT industry" were compiled;
 - A register of methodological and didactic materials for the above training courses, which will be consistently replenished according to the needs of students and teachers of Lyceum, the KNU was created;
 - A system of activities to promote the objectives of the project "IT-education" was implemented.

3 Conclusions

The effectiveness and quality of the training of professionals to meet the needs of the IT industry in Ukraine can be achieved only within a complete model of the educational process based on the interaction of education, science and business. The model should include a vision of educational outcomes (sets of substantial knowledge, specific skills and competencies), support systems (standards, curricula and methods, evaluation skills, learning environment), the mechanism of interaction between schools and universities with employers at all stages of study.

It should be noted that investments in the status are more attractive to businesses and the IT industry is more interested in increasing of the number of qualified graduates and in compliance with the reorganization of the educational process in the sector of IT education and improving its quality.

References

1. Bublik, V., Hlibovets, M., Oletskiy, O.: Ways of transformation of IT education in software engineering: experience of computer science department NaUKMA, *Naukovi zapysky*. Kyiv, NaUKMA, Computer science, vol. 73, pp. 9--13 (2007) (in Ukrainian)
2. Bublik, V., Hlibovets, M., Oletskiy, O.: Models of transformation of information education in context of movement towards the information society: experience of computer science department NaUKMA *Scientific studies. Methodological journal*. Publishing house Petro

- Mogyla State University, Mykolaiv, vol. 71, Issue 58. Pedagogical science. pp. 60--64 (in Ukrainian)
3. Government special-purpose innovation ICT program for the teaching and educational process of comprehensive schools “One hundred percent” (in Ukrainian)
 4. Oder “About the improvement of activities concerning the ensuring of the educational development in ICT area for a period until 2013”, <http://zakon2.rada.gov.ua/laws/show/1036-2011-%D1%80> (in Ukrainian)
 5. Ostrovskiy, K.: Personnel training in Ukraine: problems, prospects on the brink of the third millennium, Khmelnytskyi, p. 346 (2002) (in Ukrainian)
 6. Sharan, R.: The issue of training programs standardization for IT masters in Ukrainian high schools, http://www.intellect-invest.org.ua/pedagog_editions_e_magazine_pedagogical_science_arhiv_pn_n1_2009_st_30/ (in Ukrainian)
 7. Subprogram “Information educational environment of educational institutions in the capital” of program “Education in Kyiv. 2011-2012”. (in Ukrainian)
 8. The competent approach in modern education: World experience. Ukrainian prospects: Library of educational policy, edited by Ovcharuk O. K.I.S., Kyiv, p.112 (2004) (in Ukrainian)

Teaching Conceptual Modeling in ER: Chen Worlds

Natalya G. Keberle¹ and Ivan V. Utkin^{1,2}

¹ Zaporozhye National University, Zhukovskogo st. 66 69063 Zaporozhye Ukraine
nkeberle@gmail.com

² Prima Development Group
vanjonito@mail.ru

Abstract. Whilst algorithmic modeling is taught intensively both in school and higher education, conceptual modeling, or modeling of data to be used by algorithms, is less highlighted in the teaching curricula. However, understanding basic conceptual modeling principles plays a very important role in practice, as the cost of wrong solutions taken at the level of conceptual modeling is usually high. Tools, accelerating learning of conceptual modeling, are rare, at least freely available or mentioned in the literature. We present our work in progress – a system called “Chen Worlds” reflecting the focus on ER paradigm of conceptual modeling, describe its use cases, architecture and technical solutions undertaken.

Keywords. Conceptual modeling, entity-relationship diagram, teaching ER

Key Terms. TeachingProcess, TeachingMethodology, ConceptualModeling

1 Introduction

A course in databases and information systems, even introductory, pays attention to the architecture of a database system and to the process of a database system building, part of which is conceptual modeling. The analysis of student works in conceptual modeling for databases, particularly in ER notation [1], has envisaged a set of common mistakes, coming from either total misunderstanding of ER notation (rather seldom, nevertheless), or from misconception of specific use cases. Provision of detailed feedback on resolving misconceptions for interested students raises the syntactical validity of their ER diagrams produced essentially. Without clear understanding of the syntax of ER notation it is hard to produce conceptual models which correctly reflect a domain at the level of semantics. We acknowledge that teaching semantically correct conceptual modeling requires much more time for practice, and restrict ourselves with a modest aim – to create an easy-to-use and friendly environment to learn syntax of ER diagrams.

The paper is structured as follows: in the Section 2, we describe and classify patterns of misconception of ER notation, detected during the analysis of students' works. Section 3 presents the related work and fruitful ideas inspiring the presented

approach to teaching ER. The architecture of the system “Chen Worlds” and the use cases are sketched in the Section 4. Section 5 presents concluding remarks.

2 Common Mistakes in Understanding ER

The notation of ER uses small and simple set of basic constructs: entities, attributes (including primary, optional, multiple), relationships, weak entities and generalizations. Well known complex entities – aggregations and associations are derived from this set.

All the mistakes in understanding ER are divided into syntactic and semantic. Unsurprisingly, usage of incorrect syntax is conditioned by a poor understanding of a domain modeled. However, we have assumed that domain modeling skill requires some experience and time, and during the introductory course we can only teach some well known heuristics for semantically correct conceptual modeling.

Let’s illustrate the idea with typical examples, taken from works of students. The absence of a primary key for an entity is a syntax mistake (see Fig. 1a), whereas usage of single-valued attribute as a primary key for an entity having all other properties as multiple-valued attributes shows a semantic misconception of a domain (see Fig. 1b).

There are more ambiguous examples, like the one depicted on Fig. 1c. Non-differentiation between the values of a property (e.g. “rain”, “snow”, “hoar-frost” etc. are some values of a property “name” of an entity “Precipitation”) and composite properties (e.g. “address” is a composite property with atomic parts “city”, “street”, “house”, “apartment”, “postal code”) could be a semantic misconception.

However, under certain circumstances, namely, assuming that the values of properties “rain”, “snow”, “hoar-frost” are ranging in Boolean data type, the diagram, depicted on Fig. 1c, could be semantically correct.

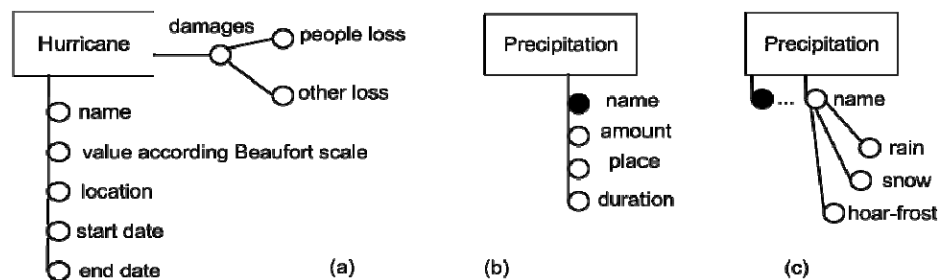


Fig. 1. Some typical mistakes in ER diagrams: (a) – no primary key; (b) – single-valued primary key with multi-valued non-key attributes; (c) – usage of attribute values as attributes.

From the syntax of ER notation and following the analysis of the students’ works in conceptual modeling, we have constructed a list of core syntactical misconceptions, a notable part of which is given below:

1. If two different entities possess exactly the same set of attributes, they could be merged.

2. Duplication of attributes across several entities is not allowed, especially if the attributes are foreign keys.
3. Primary key should be introduced for every concept in an ER diagram.
4. Primary key attributes cannot be optional, they should be mandatory.
5. Primary keys of relationships are not allowed on an ER diagram.
6. Cardinalities of a relationship are from the set $\{1, M\}$.
7. Participation cardinalities (optionality) of a relationship are from the set $\{0,1\}$.
8. A relationship cannot be directly related to another relationship.
9. An entity cannot be directly related to another concept.
10. Composite properties should have as parts only properties, not values of properties.
11. Weak entity cannot be related to strong entity with cardinalities, different from $(1,1):(1,M)$.
12. Participation cardinalities for aggregate entity and its parts are to be mandatory.
- 13.

To resolve at least these misconceptions and to help tutor and students in understanding conceptual modeling in ER at least at the level of syntax we exploit several techniques, described in the next section.

3 Accelerating Learning, Root Questions and Gaming Environment

According to the review of motivations for achievements in mathematics [2]: "...If students realize that their successes are meaningful and result both from their abilities and from a high degree of effort, they are likely to believe that they can do mathematics if they try...". Understanding conceptual modeling from our point of view is more to efforts put to understand the notation first and to apply it for different domains.

The idea of accelerating learning when teaching conceptual modeling as part of a course in databases and information systems has roots in one of the works of Professor Emeritus Jeffrey Ullman¹ at Stanford University. Prof. Ullman with colleagues had created Gradiance On-Line Accelerated Learning System² – a service for on-line training in solving simple-to-tricky exercises in various fields of Database Systems, Compilers, Automata Theory and Operation Systems.

Following the idea of a root question [3] they had introduced sets of exercises, addressing the main problems of the particular field of study, sets of possible mistakes, and sets of hints (and in some cases, solutions) to avoid those mistakes.

The idea of immersion of learning to gaming environment is widely used in practice for various fields of study. Of particular interest are the systems for teaching programming, like ALICE [4] – for object-oriented programming, PictoMir [5] and KuMir [6] – well known and respected environments aiming at teaching children and secondary school pupils basics of algorithmic thinking. The main distinguishable

¹ <http://infolab.stanford.edu/~ullman/>

² <http://www.gradiance.com/>

feature of such systems is immediate visualization of the choices a scholar makes, and consequent visualization of the solution by the system.

For example, in PictoMir there are: Environment – a part of the Universe, usually, a plain surface made of squares, Robot-Performer – a robot, able to move on that surface one step back or forth, rotate 90 degrees to the left and to the right, and Learner – usually a child, that carries out a task, e.g. “Let Robot-Performer fill in with some color all the squares at the corners of the surface”, writing an algorithm of the kind “Move 1 step forth, Fill the square, Turn 90 degrees left, Move 1 step forth...”. Algorithm writing is also replaced with picking up the proper symbol of the robotic language, like “left arrow”, “turn 90 degrees left arrow” and so forth. The solution proposed by a scholar is executed “as is” step by step, and it’s easily seen at what step of the algorithm Robot-Performer fails.

For Conceptual Modeling in ER we initially have a set of graphic primitives and a set of connection rules for primitives. Adopting the idea of a root question we propose an environment for building, visualization and validation of conceptual models in ER notation, described in the next section.

4 Chen Worlds: Use Cases, Architecture and Technical Solutions

Chen Worlds – is a cross-platform software system for learning, building, visualization and validation of conceptual models in ER notation.

We have identified two main roles of actors in Chen Worlds: a Scholar that is a person, who learns conceptual modeling for databases, and a Tutor, who prepares teaching materials.

A Scholar use case

A Scholar uses GUI to access the system. His/her aim is to obtain certain knowledge on how to use Chen notation in conceptual modeling for databases. The system proposes a Scholar a set of examples, each of which is oriented on answering of one root question.

Each example is presented as a text, shown to a Scholar, or as a picture, depending on the type of question. There are one or several correct answers (there are situations when several solutions are correct in Chen notation), which are usually not shown to a Scholar until he or she submits at least one solution. The questions are created according to the root question technique [3].

A Scholar may also use the system as a guide to notation, choosing a theme from a list of themes dedicated to conceptual modeling in ER.

A Tutor use case

A Tutor uses either GUI or usual text editor to create examples of ER diagrams. His/her aim is to construct different examples, demonstrating the applicability of a particular problem addressed with a root question. A Tutor edits a set of rules for detection of core misconceptions (see Section 2).

The architecture of the system is presented in Fig. 2. The system consists of VISUALIZER, SOLVER, PARSER and GUI components.

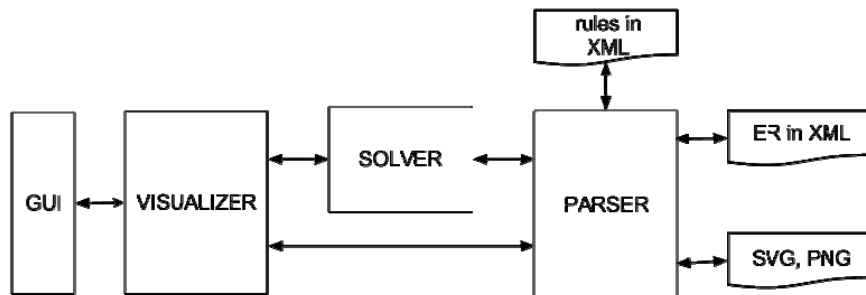


Fig. 2. Architecture of Chen Worlds system.

VISUALISER shows a task (a text, a picture, or both), shows a hint in the text of a task, and shows a hint in a picture. VISUALIZER provides a palette of graphical primitives of Chen notation.

In order to correctly present an ER diagram after reading its XML encoding, VISUALISER performs initial layout task, assigning each entity, relationship and generalization particular absolute places on a working space. Attributes, cardinalities and optionalities are placed relatively to entities/relationships they belong to.

With respect to the user role, VISUALISER may consult SOLVER and restrict the applicability of elements of the ER notation, avoiding syntactically incorrect diagrams (suitable in Tutor mode to save time). For Scholar mode VISUALISER allows arbitrary combinations of elements.

A solution submitted by a user is parsed into XML presentation and evaluated by SOLVER. SOLVER compares a solution submitted by a Scholar with respect to a correct answer of the task, detects mistakes, using a set of predefined rules, already created by a Tutor, and depending on which rules are violated, returns to VISUALISER a hint code for a user. If several mistakes are appeared in one submitted solution, SOLVER first returns hint code of highest priority (which means the most serious mistake), and in case a Scholar correctly resolves the addressed mistake, re-checks the rules again.

PARSER reads a file of a task (in XML), saves a solution (both in XML and in graphical presentation) created by a Scholar, saves a task (both in XML and in graphical form) created by a Tutor, reads a file of rules for detecting mistakes.

Examples of tasks are written as XML documents with an XML schema, substantially extending developed in [7] XML Schema with cardinalities, optionalities and generalizations. Additional XML Schema was developed for presentation of rules for detecting mistakes.

5 Concluding Remarks

Understanding conceptual modeling plays an important role in building useful and extensible database systems. There are many tools facilitating the process of construction of conceptual models (e.g. Computer Associates ERWin Data Modeler,

IBM Rational, a lot of others), but most of them just use correspondent notation (e.g. IDEF1X, Crow's Foot, UML etc.) and do not teach it. Tools facilitating the process of learning conceptual modeling are rare.

Taking classical ER notation as an example it is shown that there exist both syntactic and semantic misconceptions of the notation itself, blocking its proper usage and leading to serious problems in future database schema construction.

Chen Worlds system is currently oriented on teaching classical ER notation, however the principles of the system could be applied to other complex graphical notations, e.g. IDEF1X, UML Class diagrams, as the hardest part of their teaching is in preparation of sets of core syntactic (and in general, semantic) misconceptions.

References

1. Garsia-Molino, H., Ullman, J.D., Widom, J.: Database Systems: the Complete Book. Pearson Prentice Hall, 1203 p. (2009)
2. Middleton, J. A., Spanias, P.A.: Motivation for Achievement in Mathematics: Findings, Generalizations, and Criticisms of the Research. *J. Research in Mathematics Education*, 30(1), pp. 65--88 (1999)
3. Ullman, J. D. Gradiance On-Line Accelerated Learning Guide for Authors. <http://www.gradiance.com/downloads/auth-guide.pdf>
4. ALICE, <http://www.alice.org>
5. PictoMir, <http://www.pictomir.ru>
6. KuMir, <http://www.niisi.ru/kumir>
7. Jin, S., Kang, W.: Mapping Rules for ER to XML Using XML schema. In: Proc. 10th Southern Association for Information Systems Conference. Jacksonville, Florida, USA, 9-10 March, 2007, pp.211-216 (2007)

Training of Future Primary School Teachers for Application of ICT at Language Lessons

Inna Khizhnyak¹

¹ Slovyansk State Pedagogical University, General Batyuk st., 19,
84121 Slovyansk, Donetsk reg., Ukraine
innngen@mail.ru

Abstract. The necessity of training of the future primary-school teachers for application of information communication technologies (ICT) in their professional activity is proven in the article. The author considers the essence of the teacher's language didactic competence, reveals constituent components of the latter, and proves the urgency of the problem of introducing the future primary school teachers to the basics of electronic language didactics as a branch of education studies. On this basis the possibilities of training the students in making language skills developing electronic and multimedia course books and also in their methodically competent application at the lessons in the primary school are described. The possibilities of using the standard set of Microsoft Office programmes to achieve language and speech skills improvement are also highlighted.

Keywords: information and communicative technologies, methodology of teaching Ukrainian language, primary-school pupils, multimedia course books, electronic course books

Key Terms: TeachingProcess, ICTComponent, Development, Integration, StandardizationProcess

1 Introduction

The modern innovative processes in the Ukrainian educational system caused by all-embracing informatization of world-wide community life are based on the laws of Ukraine "About the conception of the national programme of informatization" (№ 75/98-VR of 04.02.1998) with amendments introduced according to the laws N 3421-IV (3421-15) of 09.02.2006, VVR, 2006, N 22, article 199, N 3610-VI (3610-17) of 07.07.2011), according to the law "About the fundamentals of information-oriented society development in Ukraine for 2007 – 2015" (№ 537-V of 9.01.2007) and others.

Nowadays one can observe that a considerable period (more than several decades) of introducing of information-communicative technologies into the educational reality has given certain results: there are domestic theoretical and practical works on problems of educational system informatization in Ukraine, multimedia and electronic

course books are made for pupils of various age, conventional school textbooks are being made over into electronic form, educational programmes employing school netbooks are being tested, in primary schools in particular, etc. Besides these reserves are daily supplemented with or changed by new theoretical generalizations, with results of applied researches of scientists, with electronic learning aids, etc. Thus, the process of educational system informatization is evolving and generally speaking its development can be defined as rapid.

Taking into account the constant need of all educational system stages for competent educators who understand the interests of modern pupils and are able to adjust them to the didactical aim, the leading place in the informatization of education belongs to the training of teachers and instructors with a sufficient level of informational-communicative competence in various educational branches. The primary stage of education as a basic one for the whole education of a person plays an important role in this process.

The urgency of the training of primary school teachers in the course of informatization is closely considered by modern Ukrainian scientists such as O.Bigych, I.Bogdanova, V.Imber, A.Kolomiyets, L.Morska, L.Petukhova, I.Shyman and others. In their works is emphasized the fact that “the state recognises one of the most significant condition of updating of education. That is the training and advanced training of the teaching staff and their acquirement of modern information technologies” [3, p. 2]. It is mentioned that in this regard innovative technologies and individual-oriented approach in universities contain great implicit opportunities.

All the scientists agree on the point that the only way to solve this problem is to update the educational system as a whole and the system of occupational teacher training which are “an integral unit of interrelated and interdependent constituents (social and economic, special, psychological and cultural) which have the common goal to bring up an all-round person” [3, p. 3].

The majority of researches of modern Ukrainian scientists deal with the problems of moulding the informational competence of future primary school teachers. There is certain theoretical and methodological groundwork:

- the notion of “informational culture” of a primary school teacher is defined and described in detail. The necessity of its forming for modern teachers: “Informational culture is considered as integrated personal formation which is the cause and indicator of training, is a system of attainments, abilities and skills in stating the need for information, the accomplishment of the search for the necessary information considering the whole range of information resources, picking, estimation, saving, integration, structuring and creation of new information. The necessity of forming informational culture is determined by the changes of informational resources in the educational process in universities and comprehensive schools. The network of infobases, of electronic educational and interdisciplinary connections is expanding [6, p. 3];
- the necessity of new approaches to the forming of occupational competence of teachers is proven: “Developing and improving the informatization process in the educational institutions it is essential to learn as many teachers as possible to use new ways of giving lessons applying information-communicative technologies and to introduce them into the process of creation and filling of the information medium [8, c. 33];

– approaches to the use of information-communicative technologies in the educational process are singled out: complex or partial use of ready electronic editions for educational purposes and the introduction of applied and instrumental programme tools to work out one's own learning aids etc. [1, p. 3]

At the same time the issue of forming the informational-communicative competence of future teachers in regard to the acquirement of subject methodology is yet insufficiently considered. Apparently during their study the students must receive not only general knowledge of information science and information-communicative technologies but also specialized knowledge as to how apply information-communicative technologies while teaching every single subject on the primary stage of school. These points are not yet brought to light in the Ukrainian science.

While examining the issue of introducing of information-communicative technologies into the process of teaching a foreign language, one ascertains that there exist theoretical works on this topic in Ukraine but their number is few and mainly concerning the learning of a foreign language (L.Morska, L. Kostikova and others). Moreover the majority of scientists cover these issues as a part of other ones – of a more general nature. There is a somewhat different situation in the Russian methodological science: in the recent years deep research is carried on in the field of language education (E.Azimov, M.Bovtenko, A.Bogomolov, L.Dunayeva, K.Piotrovska, E.Polat and others), of electronic language education (O.Hartzov). Owing to these researches the genres of electronic language-teaching editions of educational kind are defined and classified, the methods of their use in the educational process are worked out. However the majority of these researches concern the teaching of foreign languages or the teaching of Russian as a foreign language. Applying of information-communicative technologies as a means of stimulation of the process of learning of one's native language on the primary stage has not been yet investigated.

The aim of our article is to illustrate the necessity for a future primary school teacher to master the fundamentals of electronic language education and to demonstrate the prospects of forming his/her skills in the making of language-teaching course books for primary school pupils.

Considering the formation of language education competence of the primary school teacher-training faculty we describe it as an educational phenomenon and thus define it as an ability to organize a high-grade process of formation of primary school pupils' language and speech skills, on high-quality scientific and methodological levels. At the same time the teacher must take into account psychological and educational specific character, use different organization forms of the teaching-educational process and constantly improve himself/herself.

In the structure of this educational phenomenon we single out three groups of constituents: basic (psychological-educational, linguistic), main (linguistic-methodological and informational-communicative) and superstructure (diagnostic, acmeological). Recognising the importance of all the constituents and their role in the formation of the language education competence of a teacher we emphasize the significance of the informational-communicative constituent as the main indicator of professional ability of a modern teacher.

In the field of language teaching the formation of the informational-communicative constituent of the language education competence consists first of all in the

acquaintance of the students with the fundamentals of the electronic language education, its significance, the classification of the genres of electronic and multimedia educational and speech production, criteria of its analysis, the methods of its applying, etc.

Concerning this O.Hartsov says that “a modern teacher must not only possess the professional knowledge of his subject but also be able to apply freely didactical and methodological possibilities of new information technologies in practice. Global integration, general informatization, expansion of economic, political and cultural contacts between countries, democratization of education, migration of workers, spreading of mass media favour the forming of a unified worldwide multi-national, crosscultural, tolerant and multilingual media with new development patterns” [4, p. 34].

The scientist sees the aim of electronic language education in the integration of experience of the traditional methods of language teaching with the advantages of information technologies and as the main function of the electronic language education he considers in supplying theoretic and practical basis of language teaching under the conditions of information community. According to the successful thought of O.Hartsov, language education transforms the spontaneous process of informatization of the theory and practice of foreign language teaching into a scientific system guided by teachers- linguists [4].

The scientist names a large number of methods of electronic language teaching (object-oriented, project method, the method of visual editing, of the activation of language abilities, of information resource, of educational event and of interaction scenarios) emphasizing that the method of visual editing is of paramount importance for a future teacher. According to the estimation of O.Hartsov, application of this method allows to solve such problems of language education as:

- to produce electronic learning aids in the necessary amount independently of the specific character of a course;
- to create electronic learning aids on the base of dynamically updatable educational supplies;
- to constantly renew the present electronic learning aids according to the changes in reality, to the pupils’ requirements and the development of electronic language teaching methods;
- to create counterparts to the present electronic learning aids which favour the improvement of the education quality;
- to engage in the process of producing electronic learning aids as many teachers as possible [4, p. 149].

In the basis of this method lies the usage of widely available and well-known software which does not require specialized computer science knowledge or programming skills. Students and teachers should use this software to create electronic learning aids. We agree with O.Hartsov as to the role of the method of visual editing since the students of the primary-school teacher-training faculty have to master a large amount of various teaching materials from different branches of science for their future occupation. That is why their computer science competence develops on the most general level of operating.

Thus, the main task of training the students of the primary-school teacher-training faculty for the application of the fundamentals of electronic learning tools of

language education in the future professional activity consists in defining the most optimal set of computer programmes which allow to create educational language teaching aids, to structure them, to update them efficiently, etc. Describing the possibilities of producing electronic supplements for language and speech lessons in the primary school scientists and teachers chiefly point to the standard set of Microsoft Office programmes with the Word plug-in, WordPad, Excel, Paint, especially PowerPoint. In the list of programmes and educational courses be the resource centre “Information technologies in the language teaching” SpellMaster Word-Based Games, HotPotatoes, Filamentality and others are pointed out [9].

It is worth noticing that each of the programmes becoming more complex acquires more possibilities but in practice the most popular among the primary school teachers stands out the programme Microsoft PowerPoint. Its advantages in the fast producing of the electronic supplements for language and speech lessons, its possibilities to combine various methods of representation the language teaching material are indisputable. We suppose that the foundation of popularity of the programme Microsoft PowerPoint among teachers lies in its availability (one does not have to search for anything, to download or to study possibilities, etc.). But as for their resources the above mentioned programmes surpass PowerPoint in many issues.

Thus, the problem of choice of software for the training of future primary school teachers and for creating electronic aids for language and speech lessons lies in the orientation to the work in those programmes which they will be able to use in the future without difficulties. Here we should describe one more programme of the standard set of Microsoft Office programmes, it is Microsoft Publisher. This programme is not popular and widely used well enough.

The programme Microsoft Publisher possesses a wide range of functions – designing of advertisements, business cards, bulletins, booklets, etc. In the teacher’s work there constantly emerge a necessity to create such journalistic editions of a sufficient quality. But we consider it advisable with educational purpose to direct students on the creation of a web page in this plug-in of Microsoft Office with educational purpose. Unfortunately, the option “web page” is missing from the latest versions of the programme, they allow only to edit and to add to the examples created in the Microsoft Office Publisher 2003. That is why we give the students a pattern suggesting them to work out a fragment of an electronic course book on a definite topic using Microsoft Publisher of any version.

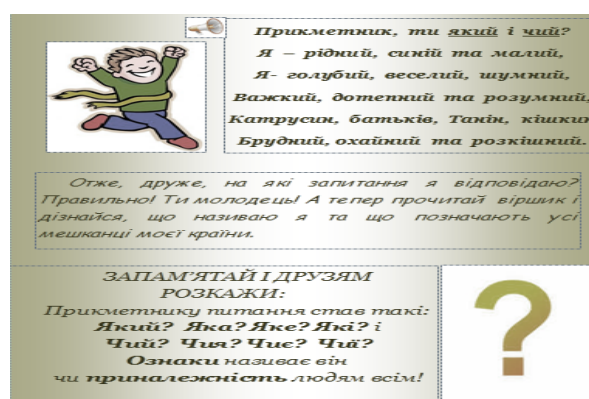
It is worth mentioning that this programme is of the same kind in its structure and means of usage as other ones in the set of programmes Microsoft Office, This allows a person without specialized computer science knowledge to master it fast, and in contrast to PowerPoint it has an important feature – interactivity, that is it gives a pupil an opportunity to navigate pages independently.

The availability of a large number of patterns and coloured schemes makes it possible to create a unique look of every single electronic learning aid for language and speech lessons and the set of elements to manage the form (switches, flags, text and edit boxes, etc.) allows to make tests to check the progress of primary school pupils using both closed and open forms of tests. The results of fulfilment can be preserved in a separate file or can be delivered per e-mail.

All this makes primary school language teaching convenient as the psychological peculiarities of primary school pupils require application of a vast number of visual

aids and the present tests often do not admit insertion of pictures, animation pictures or sounds which do not relate immediately to the task. Besides an important favourable feature of application of the programme with educational purpose we see in the possibility to preserve the result in the layout of one web page without downloading it into Internet. This web page can be opened by means of any browser.

As an example can serve an electronic course book in Ukrainian language for the fourth grade pupils on the topic “Adjective as a part of speech” made in 2011 by means of the programme Microsoft Publisher within the framework of a master’s thesis at the primary school teacher-training faculty of Slovyansk state teacher training university. This electronic course book contains five main pages: “Hello!”, “Think”, “Try”, “Your helpmates”, “Test yourself”. They contain correspondingly the main material concerning adjective as a part of speech (fig. 1), a task to select an appropriate adjective for a given animation picture, tasks to describe the characters of animated situations and to compose a text with a large number of adjectives, concise theoretical material on grammar characteristics of adjectives and their lexical categories (fig. 2), tests and other tasks to check attainments, abilities and skills of the fourth grade pupils (fig. 3).



Прикметник, ти який і чий?
 Я – рідний, синій та малий,
 Я- голубий, веселий, шумкий,
 Важкий, дотепний та розумний,
 Катрусь, батьків, Тахн, кішкин
 Брудний, охайний та розкішний.

Отже, друже, на які запитання я відповідаю?
 Правильно! Ти молодець! А тепер прочитай віршик і
 дізнайся, що називаю я та що позначають усі
 мешканці моєї країни.

ЗАПАМ'ЯТАЙ І ДРУЗЬЯМ
РОЗКАЖИ:
 Прикметнику питання став такі:
Який? Яка? Яке? Які? і
Чий? Чия? Чье? Чій?
 Ознаки називає він
 чи **приналежність** людям всім!

?

Fig. 12. A fragment of the page “Hello” of the electronic course book “The Land of Adjectives”.



Згадай і назви якомога більше мешканців моєї країни,
 які називають ознаки чи належність чогось цим
 симпатичним істотам і предметам. Тобі допоможуть мої
 запитання.

 Який?
 Чий?

 Яке?
 Які?

 Яка?
 Які?

Fig. 13. A fragment of the page “Think” of the electronic course book “The Land of Adjectives”.

Визнач рід та число іменника на малюнку і прикметників, які ти до нього дібрав.

Зроби висновок:
Рід і число прикметника залежать від

Завдання 2

Заповни таблицю: упиши поряд із прикметниками потрібні рід, число і відмінок.

Прикметник	Рід	Число	Відмінок
червона			
воєвничі			
Даринкими			
дідусевого			
шляхетному			
на українських			

Fig. 14. A fragment of the page “Test yourself” of the electronic course book “The Land of Adjectives”.

Such contents and structure of the course book allow to apply it at the lessons (fragmentary), in the individual, unassisted, home work, etc.

Consequently, the issue of informatization of education on all its stages requires urgent attention to the training of teaching staff taking into account information-communicative competence. In the field of the methods of Ukrainian language teaching of future primary school teachers we define an educational phenomenon of language education competence of a teacher and its information-communicative constituent as chief guiding lines of modern language education training. The basis of this process is the mastering of theoretical footing of electronic language education and mastering of the skills of making electronic language teaching aids depending on specific purposes of their employment.

The outlooks for further scientific researches in this field are in the detection and study of another available and user friendly software for students to create electronic language teaching supplements for language and speech lessons in primary school.

References

1. Bigych, O. B.: Information-Communicative Portfolio of a Teacher as a Means of his Professional Independence. In: Bulletin of the Vasyl Karazin Kharkiv National University. vol. 897, pp. 164 – 168. Kharkiv (2010) (in Ukrainian).
2. Bovtenko, M. A.: Computer Means of Language Teaching: Modern Opportunities/ In: Case Studies. vol. 5, pp. 25 – 37. (2011) (in Russian).
3. Bogdanova, I. M.: Professional Teacher Training of Future Teachers by Means of Innovative Technologies: Author's Abstract of Dissertation of a Doctor of Educational Sciences. Kyiv (2003) (in Ukrainian).

4. Hartsov, A. D.: Electronic Language Education in the System of Innovative Language Teaching: Author's Abstract of Dissertation of a Doctor of Educational Sciences. The Peoples' Friendship University of Russia, Moscow (2009) (in Russian).
5. Imber, V. I.: Educational Conditions of Applying Multimedia Means of Teaching in the Training of a Future Primary-School Teacher: Dissertation of Candidate of Educational Sciences. Vinnitsa State Teacher Training University, Vinnitsa (2008) (in Ukrainian).
6. Kolomyets, A. M.: Theoretic and Methodical Foundations of Information Culture of a Future Primary-School Teacher: Author's Abstract of Dissertation of Doctor of Educational sciences. Kyiv (2008) (in Ukrainian).
7. Morska, L. I.: Theoretic-Methodical Foundations of Training of Future Foreign Language Teachers for Applying of Information Technologies in Professional Activity: Author's Abstract of Dissertation of a Doctor of Educational Sciences. Ternopil (2008) (in Ukrainian).
8. Petukhova, L. E.: Theoretic-Methodical Foundations Computer Science Competence of Future Primary School Teachers: Dissertation of Doctor of Educational Sciences. Kherson (2009) (in Ukrainian).
9. Resource centre "Information technologies in the language teaching", <http://www.itlt.edu.nstu.ru/itltcourse.php> (in Ukrainian).

The Usage of Educational Portal for Distance Learning

Tatyana Zaytseva¹

¹ Kherson State University, 40 r. Zhovtnya 27, Kherson, Ukraine
sunny@ksu.kh.ua

Abstract. It has been solved in 2011-2012 years for masters of faculty of physics, mathematics and computer science within the limits of a subject "the Technique of teaching of computer science in higher educational institutions" to introduce the Distance learning and training with use the Internet-technologies. The purpose of course: to get acquainted with the systems answering to the standard IMS, SCORM; acquisition of skills of creation and use of remote courses.

Keywords. System of the distance learning, Open Source.

Key Terms. TeachingProcess, InformationCommunicationTechnology.

1 Introduction

It's mentioned in the Law of Ukraine "About higher education" that currently in institutions of higher education the distance learning is used as well as full-time, part-time and external learning. In this document the necessity of constant improvement of specialist's general and professional level of education is accented. [1, 2]. Distance learning has a powerful potential to carry out this task. It is proved by the experience of using such form of education not only abroad in the developed countries but in some Ukrainian institutions of higher education.

In modern system of education the transition from reproductive model of study to self-oriented, creative model in the center of which there's active independent cognitive activity of every individual resulting in change of all the components of the didactical system is observed.

The most prominent example is distance education which involves the implementation of up-to-date efficient learning technologies. Essentially, the distance learning presents students' organized and independent activity in mastering new branch of knowledge and using it in practice. At the same time the role of teacher in distance learning is essentially been changed and it is more encouraging the cognitive activity than declaration of knowledge.

Every educational system is based on a certain didactical conception. This fact determines the selection of content, methods, organizational forms and means of

study. We deal with a new form of education: distance learning with the use of up-to-date IT methods.

The appearance of up-to-date hi-technology educational materials and using them in the professional activity requires the certain qualification and work organization of teacher of higher school educator and mid-level teacher.

The presentation of educational material which involves the communication between the educator and students requires more active and intensive interactions among the members of the educational process. The up-to-date communication technologies give such an opportunity; however, more than usual educator's efforts are needed.

Planning and implementing the distance courses in the studying process a tutor has to be an expert not only in his/her subject field, but use IT at a high level and know the principles of computing projecting and design. So far as the technology base of courses is improving quickly, the process of planning and support of educational courses is getting more complicated. It requires special skills and pedagogical methods from the tutor. Besides, up-to-date IT also requires the certain quality of educational materials because a great number of users will have an open access to them.

Within preparation of future teachers of the higher school we tried to lead their knowledge, skills to requirements of today, namely – to prepare experts in the field of distance learning.

The purpose of this subject: studying, analysis, research of methodical and practical decisions of questions in subjects of a is natural-mathematical cycle and use of system MOODLE in training.

The essential obstacles of a successful implementation of distance learning in Ukraine are unregulated legislative and normative base of this form of education for institutions of higher education, the absence of unified standards and principles of the planning of distance courses, insufficient training of personnel and of course financial and technical support.

The problems and conditions of organization and implementation of the distance form of education were the subject for research studies of native and foreign researchers. They are Backer H., Bykov V.Y., Kukhareenko V.M., Moiseyeva M.V., Morze N.V., Oliynyk V.V., Polat Y.S., Rybalko O.V., Smirnova-Trybulska Y.M., Trius Y.V. and others.

2 Main Problem Solution

In 2011-2012 in Kherson State University the discipline “Methods and technology of distance learning” was introduced into practice for students getting the master degree in Informatics.

The main goal of the course is to form the knowledge and skills to plan and to use distance learning courses in the future professional activity.

During the course students are offered to diverge from the usual full-time attending and to try to study “in distance”, students are offered to regulate independently the time they learn new information. Students may do practical tasks at any time working

by means of Internet on the platform of the distance learning and listen to any lecture in the university. The individual schedule of educational process for each student is provided in the distance learning though all the test classes and consultations are held with educator and students interacting directly.

The goals of the course:

Methodical goals

- To form in students methodically competent in using distance learning in professional pedagogical activity.
- To expose the significance and essence of projecting didactical models, the notion of methodical educational system, its structure and implementation.
- To clarify the psychological and pedagogical aspects of learning of the main notions of professionally-oriented disciplines. To direct students towards the need and opportunities of changing the content and methods of tutoring the professionally-oriented disciplines according to the IT state of development.
- To form the knowledge and skills in the area of objective estimate and analysis of advantages and disadvantages of the distance learning, models and types of distance courses.

Cognitive goals

- To develop the ability and feeling of necessity to self-educate and to self-improve, to research the ways of improving the process of teaching of professionally-oriented disciplines.
- To develop and extend the general idea of ways and perspectives of global informatization in the area of education.
- To provide the knowledge and to form the skills in planning academic disciplines in the institutions of higher education to use the distance learning in professional activity.

Practical goals

- To form the knowledge, skills, which are necessary for the creative teaching the academic disciplines in different conditions of technical, programming and methodical support.
- To provide future educators with the knowledge and skills concerning theme planning; development of the methods for conducting lectures, practical and laboratory classes; selection interactive methods and forms of study; use of Internet for educational purposes; assessment of the results of study according to the Bologna system.
- To form the knowledge, skills of distance courses planning and support.

The course “Methods and technology of distance learning” is both attended and distance. The material of distance part of the course is introduced in the system of the distance learning KSU ONLINE, which is built on the base of the open platform Moodle. The distance course consists of 16 weeks. First and second weeks are basic where students get acquainted with the models of the distance learning, the principles of structuring the courses, its resources and elements which are built on the Moodle platform. (fig. 1)

On the 3rd-12th weeks they form the skills of planning of the distance courses and the use of up-to-date technologies of study acquainting with the lectures, additional

data resources, the experience of educators of KSU chair of Informatics, deal with the development of methodical material and creating their own distance course.

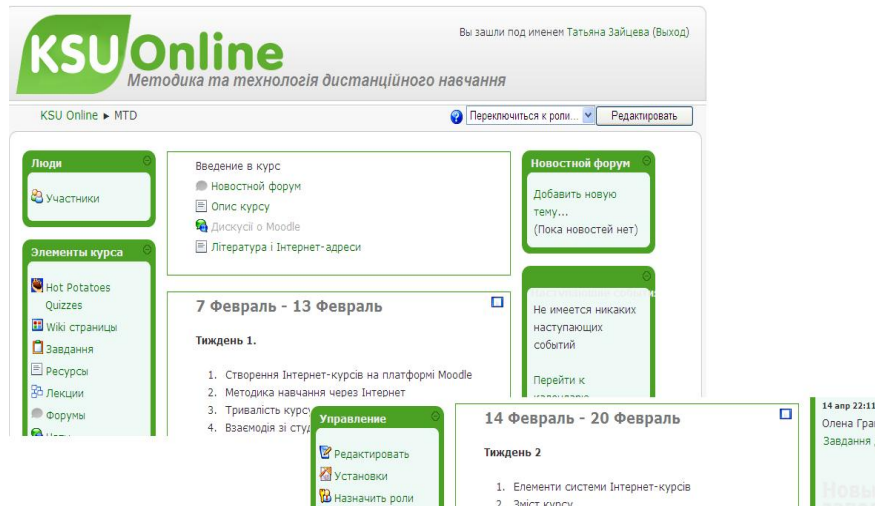


Fig 1. Distance course “Methods and technology of distance learning”.

On the last weeks students are signed up to other distance student courses which were developed on the alternative system of the distance learning. That is several studying groups are organized and students study and analyze other students’ distance courses, and in their own course they act as tutors.

Students take active part in the discussion of interesting problems on the forum and chats, and the virtual academic environment provides them with all necessary educational material.

Course study finishes in the presentation and defense of one's own distance course. They evaluate the quantity, diversity and relevance of the elements of the distance course, used by students – Lesson, Resources, Task, Exercise book, Tests, Questionnaire, Voting, Seminar, Dictionary, and synchronous and asynchronous forms of communication with students as well: Chat, Forum, Internal system of message exchange, communicator program, e-mail etc. (fig. 2)

Students were proposed the topics of distance courses, which are connected with their future professional activity. E.g. masters in specialty "Computer Science" developed the following courses: “Basic Principles of Internet Technologies”, "Basic Principles of Software Development (by example of C++)", "Data and Knowledge Bases".

Lecturers of the course “Methods and Techniques of Distant Learning” are advisors in the educational process and direct students’ academic activities, check tasks, organize interaction and communication, analyze the educational process and correct the course permanently.

See the course "Methods and Techniques of Distant Learning" and students’ distance courses on the distant learning platforms at the websites: <http://www.ksuonline.ks.ua/>, <http://www.ksu.ks.ua/dls>.

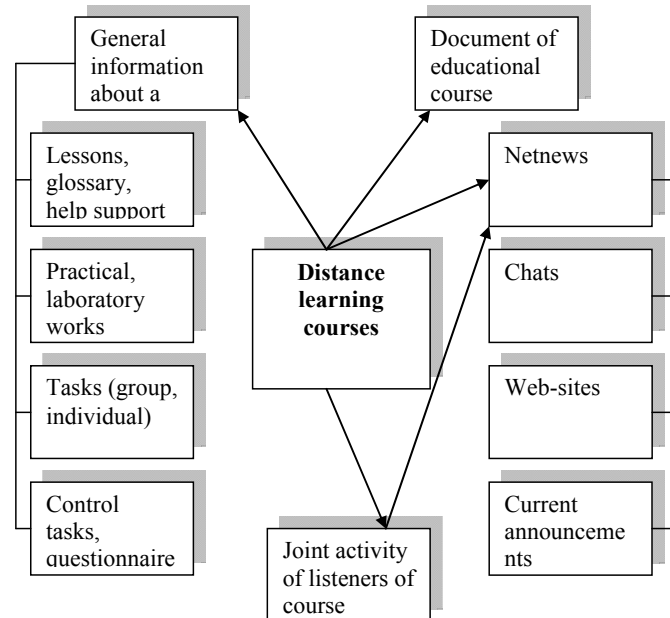


Fig. 2. Model of course of the distance learning courses.

To create their own courses students used two distant learning systems KSU ONLINE and “Kherson Virtual University”. Both systems satisfy the IMS, SCORM standards (fig. 3).

Almost all distance courses, developed by students of the specialty “Computer Science” are connected with the computer programming, that’s why the demand of methodologically based selection of the material for informational and didactic content of distance courses and testing systems caused some difficulties. Distinctly formulated standards of distance courses, which are connected with the computer programming, improving of the methodical training system of skilled staff and creation of the international distance learning platform to provide the experience exchange and researches could help to solve this problem.

Approbation of the distance courses was organized during the complex scientific pedagogical students’ practice. The main thing, that students have studied, is the methodologically based approach to the selection and using in professional activities of informational communicative technologies (notably distant learning platforms) to achieve pedagogically important results.

Distance learning system KSU ONLINE is developed on the basis of Moodle open platform. The server part of module was implemented as standard LMS Moodle mode. Moodle is a Content Management System (CMS) developed to create online courses. Such systems are often called Learning Management System (LMS) or Virtual Learning Environments (VLE).

Moodle is an instrumental environment for development both online courses and educational websites. The project contains inherently the theory of social constructivism and its using in education.

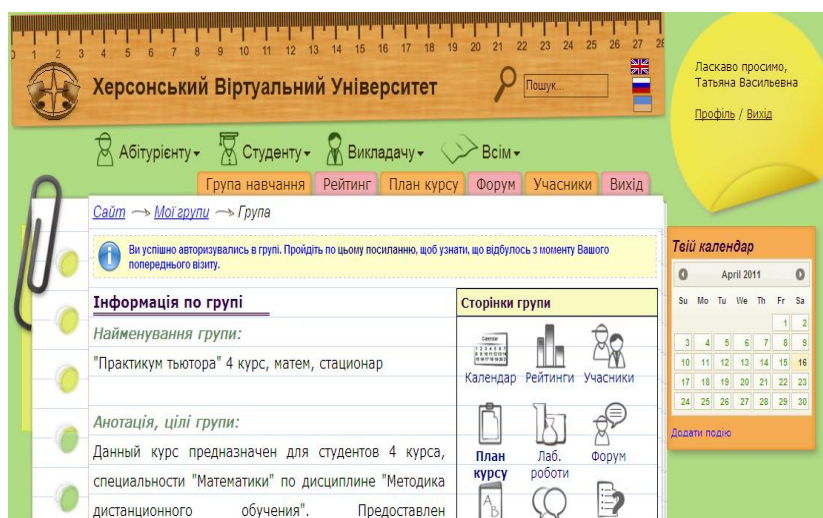


Fig 3. Distant learning systems “Kherson Virtual University”.

This free program complex meets the most e-learning demands of users by its functions, simplicity and convenience.

Moodle proposes the wide range of opportunities to support completely educational process in distant environment – different ways of giving curricular material, knowledge and progress monitoring.

Moodle is being distributed as the software with an open entry code (http://www.opensource.org/docs/definition_plain.html) under the GPL license. Moodle open program code is available on the following website: <http://www.moodle.org/> to make changes, improving, modifications, which is developed almost every day by the world community specialists in the field of the software development and education programs support.

3 The Conclusion and Ways of Further Researches

There are a lot of variants for distance learning systems, which have different technical capacities, options, spheres of using, prices, hardware demands. Among them there are commercial products, such as Oracle (i-Learning), IBM (Learning Space), WebCT, e-Learning of “Hypermethod Company” (St. Petersburg), etc and Open Source products: MOODLE, ATutor, Dokeos, Claroline.

In result of different comparisons and tests, due to didactic, organizational, technical and, especially, pedagogical reasons, users and a certain educational institution may choose and install Open Source MOODLE system under GNU/GPL license. It can support LMS, CMS and VLE at the same time (i.e. it can be used to support all project, implementation and administrative stages of the educational

process). It meets the majority of modern demands concerning distance learning support systems as well.

System administration, creation of courses and their publication by a simple web-browser interface do not require the user to have any special computer study skills, but gives possibility to give mind on the subject filling of courses and their introduction in an educational process.

Students within the limits of course "Methods and technology of distance learning" purchased knowledge and ability not only in relation to development of the controlled from distance courses from an informatics but also had the opportunity to visit a role both tutor of own course and in a role of listener of other student courses.

References

1. A law of Ukraine is «On higher education». - Kiev (2002) (in Ukraine)
2. Conception of development of the controlled from distance education is in Ukraine. Ratified Decision Department of education and science Ukraine on December, 20 in 2000. NTU «KPI», Kiev (2000) (in Ukrainian)
3. Smirnova-Tribul'ska E.M.: The Controlled from distance studies with the use of the system MOODLE. Educational and methodical manual, Publishing house Ailanthus, Kherson (2007) (in Ukrainian).
4. Zayceva T.V. Enlargement and moduleness of disciplines in teaching of informatics in the Kherson state university. Theory and method of studies of informatics, Vipusk VII: In 3-th volumes, Publishing department of NMETAU, pp. 173-176. Krivoj Rog (2008) (in Russian).
5. Moodle platform, www.moodle.org
6. Distance learning system, www.uceba.ks.ua
7. Distance learning system KSU ONLINE, www.ksuonline.ksu.ks.ua
8. Distant learning systems "Kherson Virtual University", www.ksu.ks.ua/dls

I.V Advances in Knowledge-Based and Information Systems

Selected ICTERI 2012 papers invited to UNISCON 2012

OntoElect Approach for Iterative Ontology Refinement: a Case Study with ICTERI Scope Ontology¹

Olga Tatarintseva¹, Yuriy Borue¹, and Vadim Ermolayev¹

¹ Department of IT, Zaporozhye National University,
66 Zhukovskogo st., 69600 Zaporozhye, Ukraine
tatarintseva@znu.edu.ua, yura.borue@gmail.com, vadim@ermolayev.com

Abstract. This research work is focused on the experimental validation of the OntoElect approach to ontology engineering in the case study of iterative refinement of the ICTERI Scope Ontology. OntoElect has been used for evaluating the commitment of the domain knowledge stakeholders to the ontological offering measured as positive and negative votes. The analysis of the measures allowed us answering the questions about the fitness of the ontological offering to the implicit requirements of the stakeholders as well as the completeness of the domain model with respect to their implicit expectations. The paper briefly presents the idea of OntoElect as a socially inspired approach for ontology engineering. It further describes the objectives and the set-up of our experiment. Finally it presents the results of the experiment.

Keywords. ICTERI Scope Ontology, ontology engineering, domain knowledge stakeholder, socially inspired approach, evaluation, refinement.

Key terms. SociallyInspiredApproach, KnowledgeEngineeringMethodology, SubjectExpert, SubjectDomain, Metric.

¹ This ICTERI 2012 paper has been invited for a presentation and publication at UNISCON 2012. The paper will appear in full in the post-proceedings of UNISCON 2012.

An Advanced Active Data Dictionary Based Framework for Flexible Corporate Systems¹

Maxim Davidovsky¹, Gennadiy Dobrovolsky¹, Olga Todoriko²,
and Vladimir Davidovsky¹

¹Zaporozhye National University, Center of Information Technologies,
Zhukovskogo st. 66, 69600 Zaporozhye, Ukraine

²Zaporozhye National University, Department of Information Technologies,
Zhukovskogo st. 66, 69600 Zaporozhye, Ukraine
m.davidovsky@gmail.com, gen@znu.edu.ua, o-sun@rambler.ru,
dvm@znu.edu.ua

Abstract. Today management activities pose new challenges for corporate information systems. The rate of changing and dynamics is increasing in the course of time as well as complexity of tasks to be solved that requires adequate reactions from agent of management. In such conditions information systems have to have a set of characteristics such as versatility, flexibility, and scalability to effectively address the goals. The paper presents our ongoing work at a generic framework for developing flexible enterprise information systems based on the concept of the Advanced Active Data Dictionary (AADD). The framework allows construction of flexible enterprise applications for efficient solutions in various domains and broad range of management activities.

Keywords. Corporate system, active data dictionary, software architecture, application development, framework.

Key Terms. SoftwareEngineeringProcess,
ModelBasedSoftwareDevelopmentMethodology, Development, Management.

¹ This ICTERI 2012 paper has been invited for a presentation and publication at UNISCON 2012. The paper will appear in full in the post-proceedings of UNISCON 2012.

PART II: ICTERI Workshops

II.I First International Workshop on Dynamics and Evolution in Intelligent Systems (DEIS)

Organized by: Costin Bădică, Vadim Ermolayev, and Vagan Terziyan

Foreword

It is our pleasure to offer you the section containing a paper we have selected for the 1-st instance of our International Workshop on Dynamics and Evolution in Intelligent Systems (DEIS 2012) which has been organized as a session in the technical the 8-th International Conference on ICT in Education, Research, and Industrial Applications: Integration, Harmonization, and Knowledge Transfer (ICTERI 2012) held at Kherson, Ukraine on June 6-10, 2012.

DEIS focuses on the aspects of dynamics and evolution in knowledge-based systems and technologies. World changes – so the reflections of the world in information and knowledge representations have to change adequately. Currently the correct and timely changes in knowledge representations and in intelligent knowledge-based systems become increasingly important. Indeed, intelligent systems are used more and more often in different domains, for example in those where intelligent integration of information is an essential requirement – ranging from heterogeneous sensor network data processing through the Web of things to linked data management and use. In those domains the environments of software systems and distributed information artifacts change sporadically and intensively. However, the software components are not adapted and knowledge descriptions are not changed in line with the changes in the World. Typically intelligent software systems and their knowledge are adapted or refined semi-automatically or manually and are available in a sequence of discrete revisions, which results in expanding and amplified distortion between the World and its reflection in intelligent software and knowledge representations. DEIS solicits contributions aiming at bridging the outlined gap – i.e. those helping to adapt and refine intelligent systems and their knowledge representations in dynamics, facilitating to their evolution and resulting in the artifacts that better and more adequately reflect the changes in the World.

We were especially seeking for forward looking contributions which may open the frontiers for new research directions in the future. This is why we were very selective. The paper on the Quality of an Ontology as a Dynamic Optimisation Problem we finally accepted is exactly of that genre and we are looking forward to see this promising research direction implemented in the future.

June, 2012

Costin Bădică
Vadim Ermolayev
Vagan Terziyan

Quality of an Ontology as a Dynamic Optimisation Problem

Michael Cochez and Vagan Terziyan

Department of Mathematical Information Technology
University of Jyväskylä
P.O. Box 35 (Agora), 40014, Jyväskylä, Finland
`firstname.lastname@jyu.fi`

Abstract. The Semantic Web is a proposal from the World Wide Web Consortium aimed at solving problems like data integration and application interoperability. To reach these goals several languages for the representation of semantic data have been proposed. One of the essential concepts behind semantic data is that the data is according to a certain ontology. However, the goals of the semantic web seem challenged because it seems essential for its working that ontologies are agreed upon and shared. This work-in-progress paper describes a first step to solving these problems. When an ontology is missing or only partially known a system might try to make an approximation of the missing part of the ontology. The quality of this estimated ontology will depend on the context of the application. This paper proposes the solving of a dynamic multi-objective and context sensitive optimisation problem as a way to evaluate the quality of the ontology.

Keywords: Semantic Web, Ontology features, Ontology quality, Contextual optimisation

Key Terms: KnowledgeEvolution, Intelligence, SemanticWebService

1 Introduction

The World Wide Web Consortium (W3C) regards the Semantic Web as a web of data. Currently, data is created at extremely high rate and is not available enough to end users. The Semantic Web aims “To do for machine processable information (application data) what the World Wide Web has done for hypertext” by supporting the creation of interoperable and linked data. [1] Linking data mostly happens by using shared identifiers and linking concepts by using shared ontologies.

The paper “Which Semantic Web?” [2] gives quite a critical view on many concepts of the Semantic Web. It states for instance that “Agreeing on a cataloging scheme for Semantic Web documents is a prerequisite for any sharing of semantic knowledge.” and “It is easy enough for computers to exchange data about computational abstractions such as filenames, sizes, usernames, passwords, etc. It is much harder for computers to exchange information about human-oriented

concepts such as happiness and beauty.”. These statements indicate that the Semantic Web might actually fail in its basic ideas of making decentralised information management possible. This work-in-progress paper describes the idea behind one possible approach to overcome these problems.

In order to address the first problem, it would be necessary to create a system which can make use of semantic data without having knowledge about the ontology used by the system which generated it. Therefore, it would be needed to derive the meaning from the data which another application generated. The approach proposed in this paper is that there would be an optimisation procedure which yields an ontology for an application processing semantic data. The second problem is that computer systems might not be suitable to describe human-oriented concepts. This problem has been tackled in the past by using fuzzy logic. One approach is described in [3], where a computational model which maps events and observations to an emotional state uses a fuzzy-logic representation. This paper is keeping the option of using an ontology based on fuzzy logic as one possible way of finding an ontology.

One important application of the proposed approach can be found in self-managed systems. The ‘executable reality’ approach as described in [4] is one example of a system which would benefit from the approach described in this article. ‘Executable reality’ is described in as “an extension of the (Mobile) Mixed Reality concept”. The described extension replaces part of the ‘static’ retrieval of information by computation of data using context sensitive business intelligence. When a device with such system is used at a location close by the sea concepts related to shores, harbours and seashells might become part of the active ontology. When the device is at a later time point used in a mountainous area the sea related concepts become partly redundant. If the device has a small storage capacity, the most optimal ontology will not contain these concepts any longer. A device which has, on the contrary, an abundant storage capacity but low processing power should keep the concepts stored to avoid computationally expensive changes in the ontology.

2 Optimisation

Optimisation problems are in general statements of problems where a best solution is to be found according to certain criteria and restrictions. The first paragraphs describe a few classes of global optimisation problems in order to introduce the Context-dependant Dynamic Multi-objective optimisation class in the last paragraph.

Static single-objective optimisation is considered a basic type of optimisation problems. The problem statement consists of a function which will be called f with domain D and range R . The domain of the function can be given explicitly as a set or described using constraints on a set. The set used for the range should have a total order relation \leq defined on its elements, i.e. there is a transitive, antisymmetric, and total relation on the elements of R .

Definition 1. An element $d \in D$ is optimal for a function f , i.e. $d \in \text{opt}(f) \Leftrightarrow \forall e \in D : f(e) \leq f(d)$

The class of optimisation problems described in the previous definition can be extended to a multi-objective variant by allowing the range of the function to be a set of vectors of dimension n . The range of the function f is thus $R_1 \times R_2 \times \dots \times R_n$ where we require a (strict) total order relation $<_i$ to be defined on each R_i . The domain of the function is a set D . Note that all single objective optimisation problems are multi-objective problems.

Definition 2. An element $d \in D$ is multi-objective optimal for a function f , i.e. $d \in \text{opt}(f) \Leftrightarrow \forall e \in D \setminus \{d\} : (\exists i \in [1, n] : f(e)_i <_i f(d)_i)$

Dynamic optimisation is essentially not different from solving a series of (multi-objective) static optimisation problems. The dynamism in the series is to be found in the way the function which is optimised or the constraints on the domain of the function being optimised are changing. The series can be represented by a function F , which maps the natural numbers to a set of pairs of a function and its domain. The functions in the tuples have a domain which is a subset of the total domain D the optimisation problem is working on. One way of solving the dynamic optimisation problem is by finding the optimal value for the functions for all possible values in \mathbb{N} and then selecting the maximum value. We denote $\text{opt}'(F(t))$ to be the optimisation problem with function $F(t)_1$ and constraints $F(t)_2$.

Definition 3. An element $d \in D$ is considered optimal for the dynamic optimisation of the function $F : \mathbb{N} \rightarrow (\text{functions}, \text{constraints})$ if $\forall t \in \mathbb{N} : \text{opt}'(F(t)) \leq d$.

Another possible definition follows:

Definition 4. A function sol is the solution of the dynamic optimisation problem if $\text{sol} : \mathbb{N} \rightarrow D$ and $\text{sol}(t) = \text{opt}'(F(t))$

In other words, a solution is such function which gives the optimal solution for each possible $t \in \mathbb{N}$.

Context-dependant dynamic multi-objective optimisation problems are an extension of the dynamic optimisation problems defined in the previous paragraph. In this case the domain of the function F is a series of what will be called ‘contexts’ instead of the natural numbers. The solution of the optimisation problem can be stated in a similar way as the definitions in the previous paragraphs. A solution of this type of problem indicates (multi-objective) optimal solutions in particular contexts, or analog optimal solutions over the range of all possible contexts.

3 Features of an Ontology

An application needs schemas or ontologies in order to give meaning to the semantic data it is processing. Different definitions of ontologies have been proposed. Gruber [5] defined, for instance, that “An ontology is an explicit specification of a conceptualization.”, which allows for a very broad interpretation. A

more concrete definition for description of ontologies is OWL2 endorsed by the World Wide Web Consortium. [6] There is a close correspondence (and sometimes even compatibility) between ontologies and description logic. OWL2 is for instance compatible with the description logic SROIQ. [7]

For this article, the concrete syntax of the ontology used is not of mayor importance. More important is the fact that an ontology has certain features. Examples of features of an ontology include coverage, cohesion and coupling. [8] In this article the properties which an ontology has independent of any context will be called the ‘features’ of the ontology. Some literature calls these properties ‘quality’ factors. The name quality will however be used in one of the following sections to describe the effect of features in a context. Other methods for measuring features of an ontology have been proposed by Burton-Jones et al. [9] and Yao et al. [10] and many other feature sets can be found in the literature.

4 Fuzzy Ontologies

For the representation of not exactly defined sets, fuzzy sets as described in “Fuzzy Sets” [11] can be used. The elements of a fuzzy set are members of the set according to a given membership function. This function defines for each element of the considered universe a grade of membership in the set. The grade of membership is a real value in the interval $[0, 1]$, where a value of 0 means that the element is not in the set and a value of 1 indicates that the element is in the set. Any value in between indicates up to which extend the element is a member of the set.

Calegari and Giucci used the concept of fuzzy sets to describe what they call ‘Fuzzy Ontologies’, ‘Fuzzy Description Logics’ and ‘Fuzzy-OWL’. [12] This research showed it to be possible to describe ontologies which are not exactly known by using membership functions. Bobillo and Straccia [13] did a similar work, but used OWL 2, and proposed a concrete XML syntax for the extension.

5 Ontology Evolution

In research on databases it has been noticed that the schemas which are used change over time. Ontologies have similar properties. Changes in the domain, the conceptualisation and the specification are unavoidable and the ontology has to be changed accordingly. The domain changes because the real world changes, the conceptualisation because the perspective changes and the specification changes when an ontology is to be represented in a language with different semantics and expressiveness. The whole of these changes is called ontology evolution. Ontologies and database schemas are different concepts. Firstly, the ontology itself can also be part of the data and this way the data becomes self-descriptive. Secondly, ontologies are explicitly designed for reuse in other context as the initial context of creation and are, moreover, decentralised by nature. Lastly, ontology models have, in general, a richer set of properties available for describing the

domain and quite often the border between the schema and instance data is blurry.[14]

The same article also describes concrete effects of ontology evolution on the data set. For instance removal of a class causes instances to have a less specific type, declaring classes disjoint makes instances that are in both classes invalid and defining a class as a subclass of another one adds new possible properties on the subclass.

Despite their differences, a more recent article by Hartung et al. [15] claims that ontology evolution has similar requirements as changes in database schemas evolution. Ontology evolution requires, for instance, “support for a rich set of changes, expressive mappings, update propagation to instances and dependant schemas/ontologies, versioning and user-friendly tools”. The same article compares several approaches for managing and tracking ontology evolution in terms of the above mentioned criteria.

6 Context and Quality of an Ontology

In order to talk about the quality of an ontology, one needs to take the context within which it is used into account. This statement can be supported by an example. Imagine for instance that one would say that an ontology which contains more concepts, is better than one with less concepts. This could be true in certain situations. However, if one imagines a system which only uses the concepts which are mentioned in the ontology with less concepts, the smaller ontology might be even better because it uses less memory space.

One way of defining ontological quality is described in “Data Driven Ontology Evaluation” [16]. This method uses a combination of a corpus and the ontology to evaluate the quality of the ontology. The corpus is a textual description of the ontology, which form the basis of different approaches of measuring ontologies described. The paper further elaborates on the fact that there is more as one quality aspect with regard to ontologies. Factors like price to build, maintenance and re-use are highlighted as very influential. Furthermore, the article mentions that the quality might be subjective to time, location and other contextual factors.

A recent survey on ontology evaluation tools was performed by Aruna et al. [17] This survey puts a stress on more technical demands of an ontology in a working system. Technical properties surveyed are interoperability, turn around ability, performance, memory allocation, scalability, integration into frameworks and interfaces. The ontological properties are limited to language conformity and consistency.

Research on improving the quality of an ontology by transformation operations on an existing ontology is described in [18]. The idea is that certain quality criteria can be fulfilled better by a transformation of the existing ontology into a new, but equally valid ontology. Concrete, several transformations are described to improve the ontology in terms of homogeneity, totality of properties, stability over time, and explicitness (as opposed to inferred) and uniformity in proper-

ties. The size of the ontology and simplicity of queries is, according to the same article, only assessable in a context.

One way of measuring the quality of an ontology might be to compare the ontology with one or more sets of data which should be described by the ontology. This comparison can be done using either the open-world assumption which is typically made in the semantic web or a closed-world assumption.

Quality does not have to be a one dimensional property. The quality of an ontology in a context can thus be a multidimensional property. The more complete the context for optimisation is, the fewer dimensions the quality will have. A 'complete' context will lead to a one dimensional quality.

7 Quality of Ontology in a Context as a Dynamic Multi-objective Optimisation Problem

As argued in the previous section, it is only reasonable to make statements about the quality of an ontology in a given context. If for a certain context the quality for two ontologies is given, then it is possible to make a comparison between the two ontologies in that particular context. However, quality will not always be one-dimensional and hence analog to the multi-objective class of optimisation problems, it is not always possible to say that either the first or the second ontology is better.

When it is possible to compare the quality of ontologies in a context, then it is also possible to define what it means for an ontology to be optimal in a given context. Because of the fact that there is no total order on the quality of an ontology in a context, it will be necessary to consider the search for an optimal ontology for a given context as a multi-objective optimisation problem.

Let C be the set of contexts, O be the set of ontologies and Q be the set of qualities. Now we can define the optimisation as:

Definition 5. *The function sol is the solution of the context-dependant dynamic multi-objective problem of finding an optimal ontology for a given context $\Leftrightarrow sol : C \rightarrow O$ and $sol(c) = opt'(F(c))$*

Where $F(C) \rightarrow (O \rightarrow Q)$ a function which maps the context c to a function which incorporates the context when evaluating the quality of an ontology and its associated domain.

It makes most sense to use the definition of context-dependant optimisation with a function since the system will be used in a real life setting and it is impossible to predict the optimal ontology over the whole life-time of the system at any point. It should also be noted that in any real system, the function sol can only be known partially and most likely by approximation. It is only known partially because the context is changing all the time and the future contexts are still unknown. Only an approximation can be found because no real system can react quick enough to all changes in a complex context in order to compute a new optimal ontology.

8 Future Research Directions

As said in the previous section, it is important to note that the context of the system in which the ontology is optimised will change dynamically over time. One important aspect of the context is the history of the system. The reason is that taking a new ontology into use causes a certain replacement overhead. A new ontology causing a big overhead has a lower quality in the context of the system. Cuenca Grau et al. [19] tried to reduce the cost of consistency checking of a new ontology by using the previously used ontology, i.e. the history of the system.

In this article we did not specify any concrete ontology notation to be used for this type of system. There is a need to evaluate the different possible classes of ontologies and see which properties of ontologies are affected in this type of optimisation. Depending on the type of ontology chosen, the system has different options for the development of its ontology. If the ontology would be for instance a fuzzy ontology, it might even be feasible to change only membership functions to adapt to changes in the context.

It is an open question how the system should search for an optimal ontology for a given context. Furthermore, this article refrained from defining a concrete definition of context and how it should be incorporated in the functions for optimisation. One popular choice for the incorporation of context is the use of weighted sum based methods. More research is needed to link particular features of an ontology to quality aspects, as well as how the context influences this.

Next, it seems reasonable to look whether it is possible to notice trends in the evolution of the ontology. A system could then take the trends into account when assessing a new ontology or predict resource consumption in the future.

Lastly, it is interesting to consider what would happen when two separate systems have their isolated evolutions of the ontology. Questions in that kind of situation include what should be done to align these ontologies, what to do with discrepancy between the ontologies, whether these systems can interact if they are using different ontologies, etc. . .

9 Conclusion

The aim of this article was to show initial findings to solve the problems of interoperability in the Semantic web in case ontologies are not shared among applications and how to allow these applications to work with more ‘humanised’ concepts. The first problem was reduced to a formulation of the finding of an optimal ontology in a given context in section 7. This context includes for instance the data processed by the application, the past used ontologies and constraints related to the system. The challenge of allowing humanised concepts is attempted by allowing fuzzy logic to be used in this kind of system.

This article was financially supported by the ‘Cloud Software Program’ of TiViT Oy. We would like to thank ‘Intelligent Precision Solutions and Services Oy’ and in particular Sami Helin for proposing the initial ‘Cloud Communication Channel’ business case in which this research idea was elaborated.

References

1. Carroll, J.J., Klyne, G.: Resource description framework (RDF): Concepts and abstract syntax. W3C recommendation, W3C (February 2004) <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/>.
2. Marshall, C., Shipman, F.: Which semantic web? In: Proceedings of the fourteenth ACM conference on Hypertext and hypermedia, ACM (2003) 57–66
3. El-Nasr, M.S., Yen, J., Ioerger, T.R.: Flame—fuzzy logic adaptive model of emotions. *Autonomous Agents and Multi-Agent Systems* **3** (2000) 219–257 10.1023/A:1010030809960.
4. Terziyan, V., Kaykova, O.: Towards “executable reality”: Business intelligence on top of linked data. In: *BUSTECH 2011, The First International Conference on Business Intelligence and Technology*. (2011) 26–33
5. Gruber, T., et al.: A translation approach to portable ontology specifications. *Knowledge acquisition* **5**(2) (1993) 199–220
6. Group, W.O.W.: OWL 2 web ontology language document overview. Technical report, W3C (October 2009) <http://www.w3.org/TR/2009/REC-owl2-overview-20091027/>.
7. Patel-Schneider, P.F., Motik, B., Grau, B.C.: OWL 2 web ontology language direct semantics. W3C recommendation, W3C (October 2009) <http://www.w3.org/TR/2009/REC-owl2-direct-semantics-20091027/>.
8. Ouyang, L., Zou, B., Qu, M., Zhang, C.: A method of ontology evaluation based on coverage, cohesion and coupling. In: *FSKD*. (2011) 2451–2455
9. Burton-Jones, A., Storey, V.C., Sugumaran, V., Ahluwalia, P.: A semiotic metrics suite for assessing the quality of ontologies. *Data Knowl. Eng.* **55**(1) (October 2005) 84–102
10. Yao, H., Orme, A.M., Etzkorn, L.: Cohesion Metrics for Ontology Design and Application. *Journal of Computer Science* **1**(1) (2005) 107–113
11. Zadeh, L.: Fuzzy sets. *Information Control* **8** (1965) 338–353
12. Calegari, S., Ciucci, D.: Fuzzy ontology, fuzzy description logics and fuzzy-owl. In: *Proceedings of the 7th international workshop on Fuzzy Logic and Applications: Applications of Fuzzy Sets Theory. WILF '07, Berlin, Heidelberg, Springer-Verlag* (2007) 118–126
13. Bobillo, F., Straccia, U.: Fuzzy ontology representation using owl 2. *CoRR abs/1009.3391* (2010)
14. Noy, N., Klein, M.: Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems* **6**(4) (2004) 428–440
15. Hartung, M., Terwilliger, J., Rahm, E.: Recent advances in schema and ontology evolution. *Schema Matching and Mapping* (2011) 149–190
16. Brewster, C., Alani, H., Dasmahapatra, S., Wilks, Y.: Data driven ontology evaluation. In: *International Conference on Language Resources and Evaluation (LREC 2004)*. (2004)
17. Aruna, T., Saranya, K., Bhandari, C.: A survey on ontology evaluation tools. In: *Process Automation, Control and Computing (PACC), 2011 International Conference on*. (july 2011) 1–5
18. Mostowfi, F., Fotouhi, F.: Improving quality of ontology: An ontology transformation approach. In: *Data Engineering Workshops, 2006. Proceedings. 22nd International Conference on*, IEEE (2006) 61–61
19. Cuenca Grau, B., Halaschek-Wiener, C., Kazakov, Y.: History matters: Incremental ontology reasoning using modules. *The Semantic Web* (2007) 183–196

II.II First International Workshop on Integration of Information Technologies in Economic Research (ITER)

Organized by: Vitaliy Kobets, Sergey Kryukov, and Tanya Payentko

Foreword

We would like to offer you the section including the papers we have selected for the 1-st International Workshop on Integration of Information Technologies in Economic Research (ITER 2012). The workshop is co-located with 8-th International Conference on ICT in Education, Research, and Industrial Applications: Integration, Harmonization, and Knowledge Transfer (ICTERI 2012) held at Kherson, Ukraine on June 6-10, 2012.

The necessity to make decisions at different levels of any organization, to verify economical hypotheses and usage of gained knowledge in the learning process requires the use of IT to process relevant information. Moreover, the skills for analytical information processing in decision making can only be realized effectively with the use of IT. However, a substantial proportion of economic studies is still not supported by comprehensive mathematical, information, and communication tools and technologies – thus resulting in the lack of quality in research on both micro, macro, and industry levels. The objective of ITER is to serve as an interface between the researchers in economics and the mentioned technologies and tools. More particularly, the following fields of study are in our scope:

- Enterprises and product markets
- Labor markets and social policy
- Macroeconomics, financial markets
- Public economics
- International trade and regional integration
- Mathematical and information-based methods taught in Economic curricula

Under international level research many scientist and researchers use the Ukrainian and Russian language, which greatly limits the possibilities for the world community to become familiar with published papers. A rigorous peer review process has been applied in ITER which resulted in the offered selection of the two papers out of 11 submissions

The paper on “Direct and Indirect Impact Analysis of Ukrainian Industries on Gross Output and Labor Market in Leontief Model” opens our section and has more focus on the use of formal methods and ICT in economic research. It is complemented by the paper on the “Econometric Analysis of the Factors which Determine the Choice of University Entrants”.

We believe that the workshop will not only be interesting to the attendees but also provide useful recommendations for decision-makers in organizations.

June, 2012

Vitaliy Kobets
Sergey Kryukov
Tanya Payentko

Direct and Indirect Impact Analysis of Ukrainian Industries on Gross Output and Labor Market in Leontief Model

Vitaliy Kobets¹

¹Kherson State University, Chair of Informatics, 40 roktiv Zhovtnya, 27,
73000 Kherson, Ukraine
vkobets@kse.org.ua

Abstract. In the paper are compared direct and indirect impact analysis of Ukrainian industries production on employment of labor market, profitability and gross output in 2010, it is developed recommendations for state policy to support of priority industries.

Keywords. Input-output model, industries, direct and indirect influence, gross output, final demand, externality, multiplier.

Key Terms. DecisionMaking, MathematicalModel, Industry, Macroeconomics, MatrixApproach.

1 Introduction

Input-output model (IOM) distributes gross output on intermediate and final demand. Algebraic input-output method allows simultaneously taking into account production costs of goods with consideration of technological interconnection. The matrix of technological coefficients demonstrates direct costs of industries on production, and matrix of multipliers shows complete costs of industries on production of their goods (with consideration technologically linked costs of the others industries). A difference between complete and direct costs forms indirect costs.

In article [3] the fraction of indirect costs of industry is calculated by means the language Octave [4] with the using of technological coefficients matrix and eigenvector of matrix.

For the input-output matrix is solved both direct problem (for the calculation of GDP), and indirect problem (for estimation of the gross value added) after the types of economical activities [5] with using existent restrictions (material, labor etc).

2 Problem Statement

Industry priority is determined by the state on the size of budget revenue in the state budget, by the increasing of employment, by the raising of gross output, that is after

explicit (direct) indexes, not taking into account the indirect influencing (external effect), which industry is diffuses on the other industries of country.

Purpose of this article is to define direct and indirect influence (external effect) of every industry of Ukrainian economy on the gross output of country for application of state support for priority development industries.

3 Results

At construction of interindustry balance take into consideration interdependence between separate economical industries and their links with final demand, that is represented into table 1 (x_{ij} – production output of i -th industry, that is used for manufacture of the products of j -th industry; X_j – gross output of j -th industry; Y_j – final demand on the products of j -th industry) [6]:

Table 3. Table of interindustry balance (IOB) with labor costs.

Industries-producers	Industries-consumers					Final demand	Gross output
	1	2	3	...	n		
1	x_{11}	x_{12}	x_{13}	...	x_{1n}	Y_1	X_1
2	x_{21}	x_{22}	x_{23}	...	x_{2n}	Y_2	X_2
3	x_{31}	x_{32}	x_{33}	...	x_{3n}	Y_3	X_3
.....							
n	x_{n1}	x_{n2}	x_{n3}	...	x_{nn}	Y_n	X_n
Labor cost	L_1	L_2	L_3	...	L_n	-	L

Let us to denote cost of labor for production of j -th industry good through L_j , where $j=1,2,\dots,n$ – number of industry, and output production of this product (gross output) through X_j , then direct labor cost on unit of j -th type of product (coefficient of direct labor-intensiveness) are determined by a formula:

$$t_j = \frac{L_j}{X_j}, j=1, \dots, n. \quad (2)$$

A general needs in labor resources for n industries is determined after a formula:

$$L = \sum_{i=1}^n L_i = \sum_{i=1}^n t_i \cdot X_i = t_1 \cdot X_1 + t_2 \cdot X_2 + \dots + t_n \cdot X_n \quad (2)$$

From a formula $X = B \cdot Y$ (B - matrix of multipliers or complete costs) we will get the following values of industries production output:

$$\begin{aligned}
 X_1 &= b_{11} \cdot Y_1 + b_{12} \cdot Y_2 + \dots + b_{1n} \cdot Y_n, \\
 X_2 &= b_{21} \cdot Y_1 + b_{22} \cdot Y_2 + \dots + b_{2n} \cdot Y_n, \\
 &\dots\dots\dots \\
 X_n &= b_{n1} \cdot Y_1 + b_{n2} \cdot Y_2 + \dots + b_{nn} \cdot Y_n.
 \end{aligned}
 \tag{3}$$

Then labor cost can be presented as follow:
 $L = t_1 \cdot (b_{11} \cdot Y_1 + b_{12} \cdot Y_2 + \dots + b_{1n} \cdot Y_n) + t_2 \cdot (b_{21} \cdot Y_1 + b_{22} \cdot Y_2 + \dots + b_{2n} \cdot Y_n) + t_n \cdot (b_{n1} \cdot Y_1 + b_{n2} \cdot Y_2 + \dots + b_{nn} \cdot Y_n)$

We will rearrange the elements regarding to final demand on the industries products, we will obtain:
 $L = Y_1 \cdot (b_{11} \cdot t_1 + b_{21} \cdot t_2 + \dots + b_{n1} \cdot t_n) + Y_2 \cdot (b_{12} \cdot t_1 + b_{22} \cdot t_2 + \dots + b_{n2} \cdot t_n) + Y_n \cdot (b_{1n} \cdot t_1 + b_{2n} \cdot t_2 + \dots + b_{nn} \cdot t_n)$

From the last equation we will calculate the complete labor costs:

$$\begin{aligned}
 T_1 &= b_{11} \cdot t_1 + b_{21} \cdot t_2 + \dots + b_{n1} \cdot t_n, \\
 T_2 &= b_{12} \cdot t_1 + b_{22} \cdot t_2 + \dots + b_{n2} \cdot t_n, \\
 &\dots\dots\dots \\
 T_n &= b_{1n} \cdot t_1 + b_{2n} \cdot t_2 + \dots + b_{nn} \cdot t_n.
 \end{aligned}$$

We will rewrite indirect labor costs as a matrix by the following formula:

$$T_{1 \times 1} = t_{1 \times 1} \cdot B_{1 \times 1}, \tag{4}$$

$$B_{1 \times 1} = (E - A)_{1 \times 1}^{-1}$$

where

- t_i - coefficients of direct labor cost maintenances in i-th industry, persons/1000hrn.;
- T_i - coefficients of complete labor cost maintenances in i-th industry, persons/1000hrn.;
- L_i - employed workers in industry, persons/1000hrn.;
- $t_{1 \times 1}$ - matrix of direct labor costs;
- $T_{1 \times 1}$ - matrix of complete labor costs;
- $B_{1 \times 1}$ - matrix of multipliers of input-output model;
- A - matrix of technological coefficients of IOM;
- E - unitary matrix.

Difference between direct t_i and complete T_i labor costs consists in that direct labor costs show an additional (marginal) needs in labor resources for i-th industry after unit increasing of final demand on the products of i-th industry, and the complete labor costs show an additional (marginal) needs in labor resources for all n industries after the unit increase of final demand on the products of i-th industry.

On the basis of direct and complete labor coefficients are developed interindustry labor costs balances for available labor resources. These balances are built as matrix models, and all indexes in them (interindustry links, final product etc) are expressed in labor measurer (amount of persons).

For reduction of matrix model dimension without the loss of generality of its conclusions we reduced the number of Ukrainian industries from 15 to 11 by regrouping, after that we yield:

List of industries:

1. Agriculture, hunting, forestry, fishing, fish-farming.
2. Mining industry, processing industry, production and distributing of electric power, gas and water.
3. Building.
4. Trade; repair of cars, common manufacture and individual commodity, activities of hotels and restaurants.
5. Activity of transport and communication.
6. Financial activity.
7. Operations with the real estate, lease, engineering and rendering of services for entrepreneurs.
8. Public administration.
9. Education.
10. Health protection and social service.
11. Public and individual service; culture and sport activity.

Values of table 2 columns for the gross output and final demand are got on the basis of Statistics state committee data [1], employment after industries is received from statistical collection [2], and the gross value added is calculated by the means of MS Excel on the basis of IOM [6].

Table 4. Macroeconomic indexes of Ukraine for 2010 year.

Number of industry	Gross output X, millions of UAH1	Final demand, millions of UAH2	Gross added value, millions of UAH	Number of employees L, thousand of persons
1	228103	109580	82431	3152,2 (16%)
2	1848780	891629	782256	3546,9 (18%)
3	104137	98183	37751	966,2 (5%)
4	324780	25348	183795	4729,1 (23%)
5	254345	116333	155914	1387,9 (7%)
6	55786	4346	31005	351,4(2%)
7	142802	30816	60396	1148,9 (6%)
8	10220	180	-10476	1078,6 (5%)
9	2221	1042	-26018	1698,4 (8%)
10	5236	2255	-16910	1348,1 (7%)
11	24667	7150	6718	783,8 (4%)

The structure of employment for each industry is presented on fig.1 (numeration corresponds previous list of industries above).

¹ Gross output consists of intermediate product and final demand

² Final demand includes: final consumer consumption, gross investments and net export

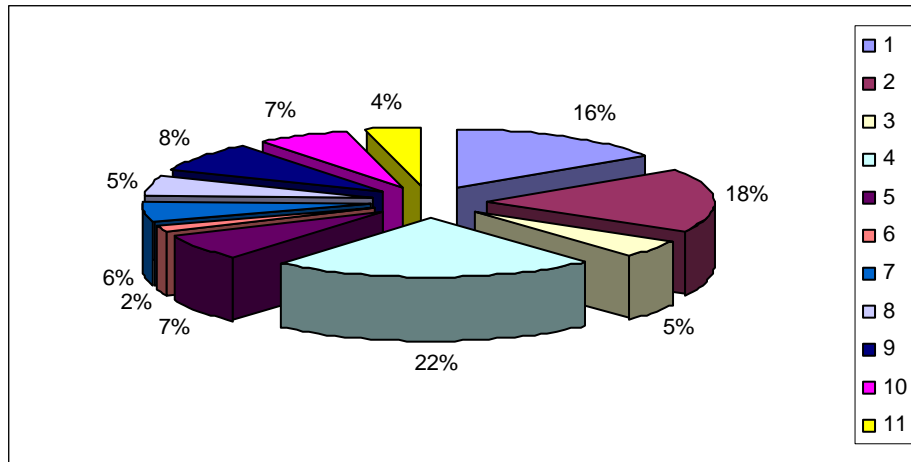


Fig. 15. Structure of employment for Ukraine industries in 2010.

By means of technological coefficients $A = \{a_{ij}\}$, calculated on the basis of table 1, we will get the indexes of profitability of each from 11 industries (table 2):

Table 5. Description of Ukrainian industries.

Number of industry	Profitability of industries, %	Multipliers of industries	Part of products for final demand
1	36,1%	2,45	48,0%
2	42,3%	2,27	48,2%
3	36,3%	2,45	94,3%
4	56,6%	1,99	7,8%
5	61,3%	1,91	45,7%
6	55,6%	1,93	7,8%
7	42,3%	2,35	21,6%
8	-102,5%	7,13	1,8%
9	-1171,5%	61,11	46,9%
10	-323,0%	11,13	43,1%
11	27,2%	2,92	29,0%
Average	43%	x	x

We got from the indexes of profitability (fig. 2), there are three unprofitable industries in 2010, which give such public benefits as state administration, education and health protection. These industry require considerably greater financing on own activities, than they earn. However finally to make decision about their influence on an economy as a whole it is possible only taking into account externality, i.e. their influence on the other industries of Ukrainian economy.

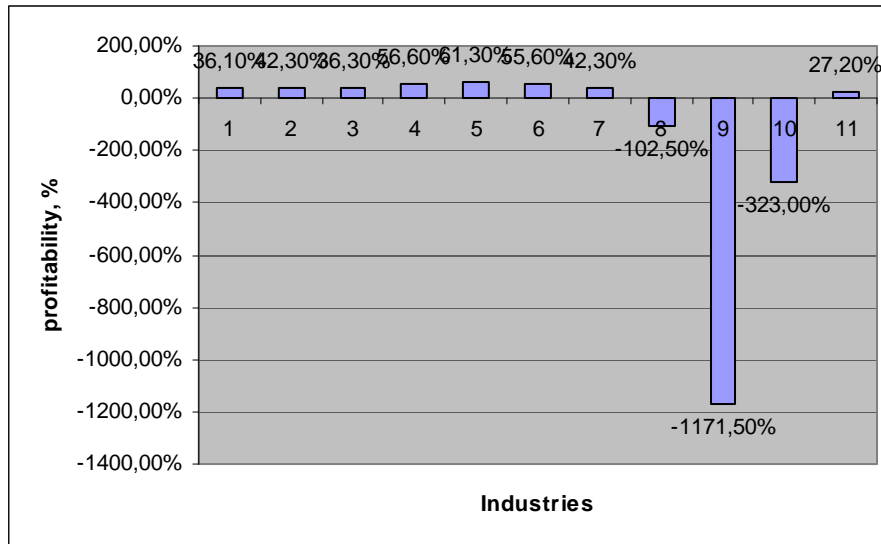


Fig. 16. Profitability of Ukrainian industries in 2010.

After calculation of industries multipliers from matrix B we will get that a most increase of GDP is induced exactly by those industries which are financed from the state budget. Every hryvnya invested in state administration; education and health protection is enhanced gross output in Ukraine on 7 hrn.; by 61 UAH and 11 UAH accordingly, that with surplus covers direct costs of state on these industries activities. Educational industry has most significant influencing and there is reserve for the increasing of country gross output. Education stimulates a positive GDP dynamics. So after a direct effect this industry on every invested hryvnya gets a loss in size of 10 to UAH, and after indirect get profit in 26 UAH (multiplication of average profitability on multiplier), that together gives the positive influencing in 16 UAH on the gross value added of Ukrainian industries.

Thus the crisis phenomena in the economy of country are not to stipulate the cost decreasing after all directions in same proportion. The state adjusting of industry has to base on that return which is had by every industry and proportionally to its influence on an economy not to decrease, and in the some cases, taking into account their external effects, to increase state costs for them for stimulation of gross output of all linked industry.

After finding relation of final demand to the gross output X (from table 1) we will get what measure of each industry is oriented into final, and which is on intermediate demand (fig. 3). We have, that build industry (3) is most oriented on final demand (94,3% whole product), and • trade industry, activity of hotels and restaurants (4); • financial activity (6) and • state administration (8) are most oriented on intermediate demand (100%-7,8%=92,2%, 92,2% and 98,2% accordingly).

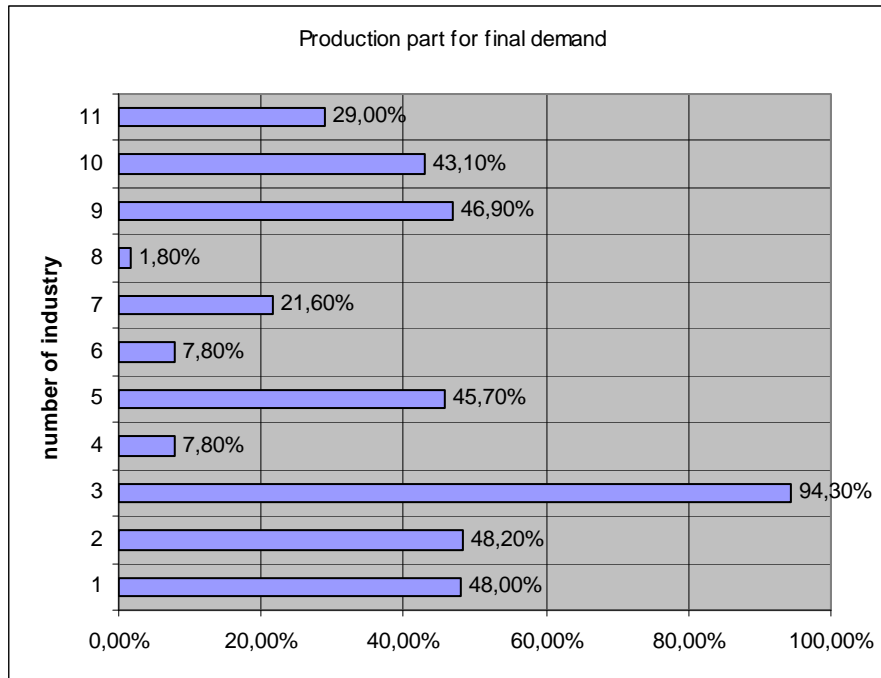


Fig. 17. Product part of Ukrainian industries for final demand.

We will consider application of interindustry balance method for direct and indirect influence on Ukrainian industries' production on a labor market. The important analytical possibility of this method is determination of direct and complete labor cost on product unit and development the recommendations in relation to state support of priority development industries for the stimulation of employment growth of country.

After the calculations after formulas (1) and (4) we will present results as table 4:

Table 6. Labor cost indexes analysis for products' manufacturing by Ukrainian industries in 2010.

Industries	Direct labor cost coefficients	Complete labor cost coefficients	Complete / direct ratio T/t	Indirect labor cost coefficients T-t
1	0,014	0,027	1,921	0,013
2	0,002	0,011	5,587	0,009
3	0,009	0,017	1,860	0,008
4	0,015	0,023	1,564	0,008
5	0,005	0,012	2,254	0,007
6	0,006	0,013	2,096	0,007
7	0,008	0,019	2,388	0,011
8	0,106	0,193	1,828	0,087

Industries	Direct labor cost coefficients	Complete labor cost coefficients	Complete / direct ratio T/t	Indirect labor cost coefficients T-t
9	0,765	2,095	2,740	1,331
10	0,257	0,330	1,281	0,072
11	0,032	0,058	1,810	0,026

We will analyze indexes above. Agriculture and fishing industry of (1) during production of own goods increases whole labor cost almost twice in comparison with its own direct labor cost. More, than twice the indirect labor cost are increased in such industries as a transport and communication is in 2,25 times (5), financial activity at 2,1 times (6), operations with the real estate in 2,4 times (7), education in 2,74 times (9).

Indirect labor cost increases at 5,6 times in processing industry (2). It means that this industry is priority so it has a primary importance for stimulation of employment in all industries of Ukrainian economy.

The least stimulation of indirect labor cost (due to growth of industry output) is observed in industry of health protection and social service (10).

4 Conclusions

Taking into account obtained influence of each industry gross output for determination of state policy strategy it follows to take into consideration results of indirect industries influencing on stimulation of employment, gross output and profitability of both individual industry and whole economy. For example, education industry after the direct influencing was unprofitable, and after indirect was profitable, so its net gross value added was positive.

Further it is planned to explore and compare the direct and indirect influencing of Ukrainian industries activity and Countries of Independent States for development of effective state policy proposals in the field of national economy.

References

1. State Committee of Statistics, <http://ukrstat.org/uk/> (in Ukrainian)
2. State Statistics Committee of Ukraine, <http://www.ukrstat.gov.ua> (in Ukrainian)
3. Stetsyok P.I., Bondarenko A.V.: About spectral properties of Leontieff model. Theory of optimal decisions. 10, 84--90 (2011) (in Russian)
4. Program Language Octave, <http://www.octave.org>
5. Kulik V.V.: Scientific activity in the system of economic circulation. Economic-mathematical design of the socio-economic systems. 4, 45--67 (2009) (in Ukrainian)
6. Kolemaev V.A.: Matematical Economics. UNITY, Moscow (1998) (in Russian)

Econometric Analysis of Factors which Determine a Choice of Entrants

A. Khristenko¹ and A. J. Weissblut¹

¹Kherson State University, Chair of Informatics, 40 rokiv Zhovtnya, 27,
73000 Kherson, Ukraine
veits@ksu.ks.ua

Abstract. In this article the factors determining a choice by entrants of a speciality and university are investigated. They include the correlation analysis of factors and an estimation of the statistical importance of the received results. The important purpose of research is the division of respondents on more homogeneous groups. Thus we have tried to estimate the factor of influence of parents on decision-making. This work is focused not only on direct use, but also on application in the educational process. Its full realization is supposed in a subsystem "Analysis" of a site of Faculty of physics, mathematics and informatics, developed in the Kherson state university.

Keywords. Factor, statistical, econometric, decision-making, university, speciality.

Key Terms. Research, Management, Model, KnowledgeManagementProcess, KnowledgeManagementMethodology, MathematicalModeling.

1 Introduction

In this article the factors determining a choice by entrants of a speciality and university are investigated. This work is focused not only on direct use, but also on application in the educational process. Really, the mentality, fixed in distributions received in this article, as a rule, does not undergo significant changes during study. Formally these distributions determine aim functions for optimization tasks, used for support of decision-making. Their realization is supposed in a subsystem "Analysis" of a site of Faculty of physics, mathematics and informatics, developed in the Kherson state university.

2 Results

2.1 Factors Determined a Choice of a Speciality

Further the histogram for distribution of the factors which have determined a choice of a speciality is shown. This is distribution for "typical" student of first year study of a speciality "informatics" (the exact sense of his typicalness is mentioned below).

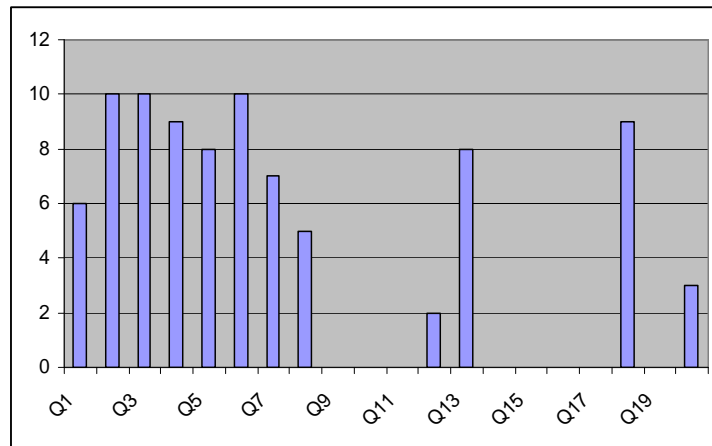


Fig. 18. Answers distribution.

Here

Q1. Reputation (prestigiousness).

Q2. I like to program.

Q3. I like to communicate with a computer.

Q4. I liked at school 1) computer science or 2) programming or 3) mathematics or 4) economy.

Q5. It in the future will provide highly paid work.

Q6. It in the future will ensure the job.

Q7. The big number of budgetary positions on this speciality.

Q8. Ease of receipt.

Factors of influence:

Q9. The type of preparation at school.

Q10. Familiars at faculty (students or employees).

Q11. The program of an exchange with foreign universities.

Q12. Advice of parents.

Q13. Advice familiars of parents.

Q14. Advice of friends.

Q15. Advice of teachers.

Q16. Speech in a class of teachers of the faculty.

Q17. Advertising of faculty.

Q18. Impression from probationers from the university.

Q19. Data from newspapers.

Q20. Data from the Internet.

1) These factors have been chosen in result of "brainstorming" where as experts students of first and fourth year study of a speciality "informatics" acted. This expert interrogation has been constructed by a technique of " six thinking hats " E. Bono [1], which provides the maximal openness and relaxedness of participants. In all cases the opinion has unanimously been expressed, that the given set of factors is full and fair. Then students of first year study of specialities "informatics" and "program engineering" has been interviewed under such essential factors. The respondents estimate the importance for them each factor points from 0 (at its full insignificance) up to 10. He arbitrarily sets a name of the file containing his interrogation (i.e. his key). The volunteer - a participant of interrogation - collects all files in a folder and sorts them (i.e. shuffles). Only after that the folder was transferred to the senior student, who conducts the interrogation: this simple and open procedure guaranteed to participants anonymity of interrogation. 3) Results of interrogation then will be worn out in table of a database of a site of faculty. The queries realizing now on the basis of pattern MVC (Model-View-Controller) [2] for a database control system MySQL give out results of the econometric analysis of interrogation. They include the correlation analysis of factors and an estimation of the statistical importance of the received results (see [3]).

Let's show some results of the analysis. The histogram of distribution of average values of factors is resulted below.

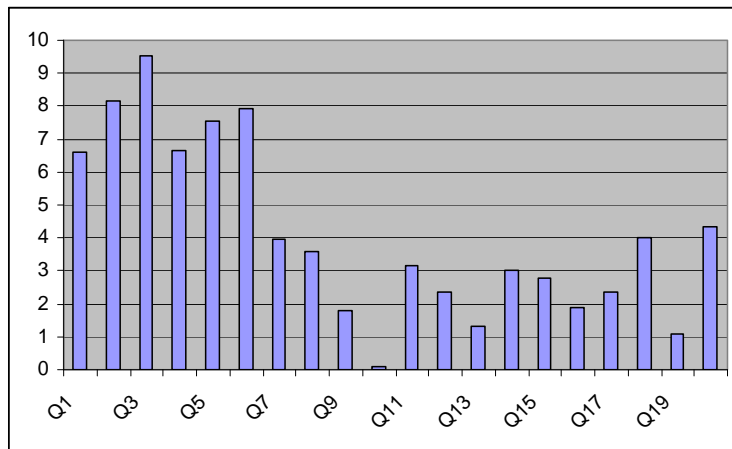


Fig. 19. Answers distribution.

1. The most significant factors appeared (in decreasing order) Q3, Q2, Q6 and Q5.
2. The most essential factors of influence on a choice with a significant separation from the others appeared (in decreasing order) Q20 and Q18. However both factors have the big dispersion: are essential to one and are indifferent for others.
3. The understanding has cleared up with respondents of the factor Q1 (reputation, prestigiousness). This factor strongly correlates only with two: Q5 and first of all

with the big separation Q6. Those respondents, who appreciably reacted to prestigiousness, reacted also to factors of influence Q18 and Q20 more essential.

If carry out the correlation analysis also between answers of respondents thus it is possible to reveal the most typical respondent (his interrogation is submitted above on the first histogram). This is that respondent at whom the average factor of correlation with other respondents is maximal.

2.2 Factors Determined a Choice of a University

Simultaneously and in the same way carried out research of mental factors and the factors of influence determining a choice of university by entrants. The histogram of distribution of the average importance of the factors, which have determined a choice of university, is resulted below.

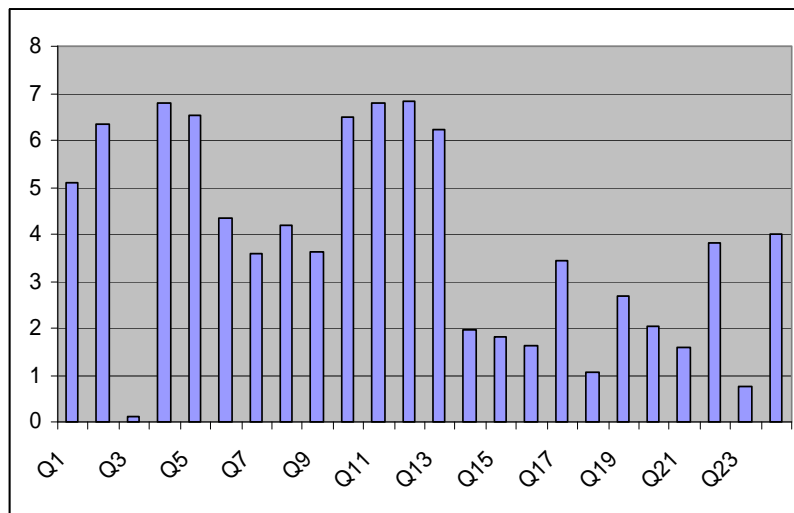


Fig. 20. Answers distribution.

Here

Q1. Distance from the university.

Q2. Reputation (prestigiousness).

Q3. Familiars at faculty (students or employees).

Q4. Teachers of this university really work with students and give real knowledge.

Q5. The student's life.

Q6. The big number of budgetary positions.

Q7. Comparative cheapness of study in comparison with other universities.

Q8. The program of an exchange with foreign universities.

Q9. An opportunity of training together with foreign students.

Q10. 4 days of training (presence of methodical day).

Q11. Wi-Fi.

Q12. Warm auditoriums.

Q13. Trainings in one building (" it is not necessary to run on buildings ").

Factors of influence:

- Q14. The type of preparation at school.
- Q15. Advice of parents.
- Q16. Advice familiars of parents.
- Q17. Advice of friends.
- Q18. The book “ High schools of Ukraine ”.
- Q19. Advice of teachers.
- Q20. Speech in a class of teachers of the university.
- Q21. Advertising of the university.
- Q22. Impression from probationers from the university.
- Q23. Data from newspapers.
- Q24. Data from the Internet.

Here results of the correlation analysis of the factors, determining a choice, turned out such.

1. The most significant factors with a great separation from the others appeared: Q2, Q4, Q5, Q10 - Q13 (conditions for study). Distinctions of the importance between these factors the tenth shares of a point, it can be neglected in comparison with root-mean-square deviations of these factors.
2. The most essential factors of influence on a choice of university is the same, as at a choice of a speciality: it is Q24 (Internet) and Q22 (probationers of university at schools).
3. At a choice of university prestigiousness (reputation) was included in number of major factors. As the correlation analysis has shown, prestigiousness is strongly connected to all significant factors specified in item1, both factors of influence from item 2, and poorly connected to other examined factors.
4. Interrogation has shown that the significant majority at first chooses a speciality and then university: a ratio of points approximately 7 : 3.

2.3 Factor of Influence of Parents on Decision-Making

The important purpose of statistical research is the division of respondents on more homogeneous groups. In this case, however, results appeared very homogeneous for factors at a choice of a speciality (with high coefficients of mutual correlation) and a little less homogeneous at a choice of university. In the latter case we have tried to estimate the factor of influence of parents on decision-making.

3 Conclusions

A direct question the significant majority have answered, that the decision they accepted independently or with small participation of parents: a ratio of points $\approx 8,5 : 1,5$. We have separated answers of 14 respondents from 21, asserting that their decisions were accepted practically without influence of parents (a ratio of points 10 : 0 or 9 : 1) and have compared answers of this group to answers of an additional subgroup. The most significant factors and essential factors of influence appeared all same in both subgroups. At the same time, the ratio of the importance of these factors

has changed. In a subgroup of the respondents asserting about independence of parents, the importance of factor Q4 has increased on a point in comparison with the data for all group. In result this factor (teachers of this university really work with students and give real knowledge) became leading (mean mark $\approx 7,57$, a mean score on all group $\approx 6,85$). In other subgroup the importance of this factor has accordingly decreased (mean mark $\approx 5,28$).

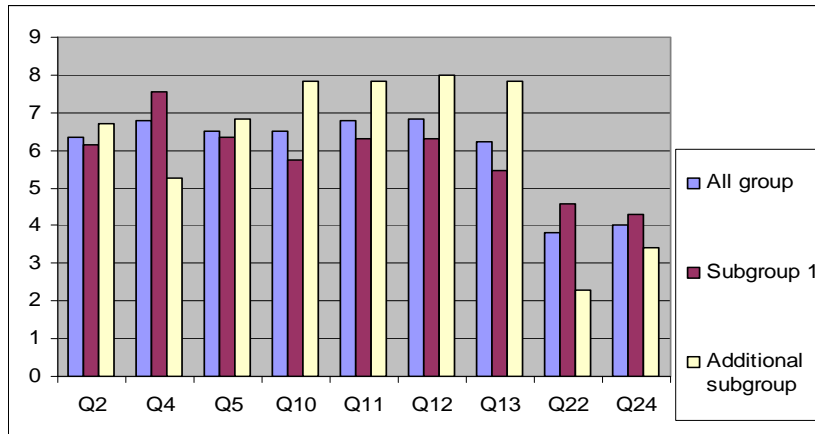


Fig. 21. Answers distribution.

At the same time, in this subgroup the importance of factors of conditions of training Q10 - Q13 have grown on a point in comparison with the data for all group. So Q13 in second subgroup $\approx 7,83$, for all group $\approx 6,21$, for independent of parents respondents $\approx 5,46$. The factor of influence Q22 (probationers of university) for independent of parents respondents $\approx 4,57$, for all group $\approx 3,8$, for second subgroup $\approx 2,28$. Mean scores of all other factors do not differ essentially in both subgroups. The comparative histogram of distribution of significant factors for subgroups is resulted below.

References

1. De Bono E.: Six Thinking Hats. Penguins Books, Boston (1997)
2. Django Book, <http://www.djbook.ru>.
3. The Social Science Computing Cooperative: Providing Computer Services for the Social Sciences at UW-Madison, <http://www.ssc.wisc.edu>.

II.III Workshop on Algebraic, Logical, and Algorithmic Methods of System Modeling, Specification and Verification (SMSV)

Organized by: Martin Strecker, Mykola Nikitchenko, and Vladimir Peschanenko

Foreword

It is our pleasure to offer you the selection of papers for the International Workshop on Algebraic, Logical, and Algorithmic Methods of System Modeling, Specification and Verification (SMSV 2012) which has been co-located with the 8-th International Conference on ICT in Education, Research, and Industrial Applications: Integration, Harmonization, and Knowledge Transfer (ICTERI 2012) held at Kherson, Ukraine on June 6-10, 2012.

Workshop SMSV 2012 is a successor of the International seminar on Specification and Verification of Hybrid Systems (SVHS) which was held in Kyiv (Ukraine) on October 10-12, 2011. The SVHS seminar was organized by Taras Shevchenko National University of Kyiv and Paul Sabatier University of Toulouse within the framework of the cooperation agreement between the universities. The seminar attracted scientists from Austria, France, Israel, and Ukraine. Presented papers demonstrated the interest in the topics on different formal methods of system development, and it was proposed to organize such seminars on a regular basis.

The scope of the SMSV Workshop was extended in order to cover the main phases of software system development. We hope that presentations and discussions will help to identify topics of mutual interest that can be considered as a base of project proposals to be submitted to international scientific programs.

June, 2012

Martin Strecker
Mykola Nikitchenko
Vladimir Peschanenko

A Case Study in Combining Formal Verification and Model-Driven Engineering

Selma Djedai¹, Mohamed Mezghiche², and Martin Strecker¹

¹ IRIT (Institut de Recherche en Informatique de Toulouse)
Université de Toulouse, Toulouse, France

² LIMOSE, Université de Boumerdès
Faculté des Sciences, Boumerdès, Algeria

Abstract. Formal methods are increasingly used in software engineering. They offer a formal frame that guarantees the correctness of developments. However, they use complex notations that might be difficult to understand for unaccustomed users. It thus becomes interesting to formally specify the core components of a language, implement a provably correct development, and manipulate its components in a graphical/textual editor.

This paper constitutes a first step towards using Model Driven Engineering (MDE) technology in an interactive proof development. It presents a transformation process from functional data structures, commonly used in proof assistants, to Ecore Models. The transformation is based on an MDE methodology. The resulting meta-models are used to generate graphical or textual editors. We will take an example to illustrate our approach: a simple domain specific language. This guiding example is a Java-like language enriched with assertions.

Keywords. Model Driven Engineering, Model Transformation, Formal Methods, Verification

Key Terms. Mathematical Model, Specification Process, Verification Process

1 Introduction

Domain Specific Languages (DSL) have conquered many different aspects of computer science. They are used in different fields such as aerospace, web-services, multi-media, etc. [1]. Certain DSLs define their semantics in natural languages. However, even though these tend to be quite easy to understand, they usually suffer from incompleteness in some cases and ambiguity in others. Therefore, there emerges a need for defining the formal semantics of DSLs in a mathematically founded framework using proof assistants. Such a phase consists in defining the abstract syntax of a DSL and then grafting a semantics on top of it, using well-understood mechanisms like structural recursion or inductive

relations. Such a semantics is often not executable, but other elements of a formal development are, such as compilers or static analyses whose correctness is proved on the basis of the formal semantics.

Interactive proof assistants such as Coq [2] or Isabelle [3] often use paradigms stemming from functional programming (type systems, function definitions), but they are as such not a programming language. It is however possible to export the formal development to programming languages such as Caml [4] or Scala [5]. A formally verified compiler, for example, can therefore be effectively executed in a standard programming language.

In order to improve the user interface for interacting with a DSL, we aim at a textual or graphical concrete syntax as provided, for example, by the Eclipse Xtext or GMF environments. Frequent changes of the DSL during the design phase make it necessary to adapt this interface easily and to re-generate it automatically, as far as possible.

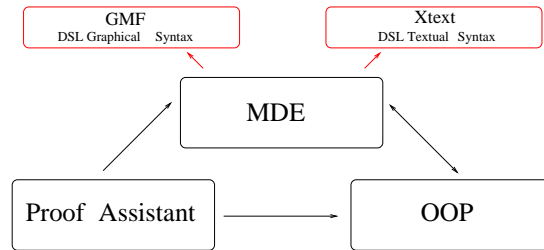


Fig. 1. Meta-modeling(MM), Verification environment and OO languages

This paper studies the interplay of these formalisms (see Figure 1), and thus constitutes a first step towards using Model Driven Engineering (MDE) [6, 7] technology in an interactive proof development. The guiding example (see Section 3) is a Java-like language enriched with assertions developed by ourselves for which no off-the-shelf definition exists. This “meta-model” (in MDE parlance) is sufficiently complex to illustrate the method and to be a case study of realistic size for a DSL. However, its formal model can be entirely defined as an inductive datatype (and this is so for most formally defined languages). In this case study, we can therefore not demonstrate some aspects of our work, such as the translation of genuine graph structures that go beyond instances of inductive data types.

Section 2 constitutes the technical core of the article; it describes a translation from data models in the functional programming world, used in verification environments, to meta models in **Ecore**: the core language of the Eclipse Modeling Framework. We illustrate the methodology in Section 3 with a case study. In Section 4 we compare our work to other approaches, before concluding in Section 5 with perspectives of further work.

2 From Datatypes to Meta-Models

In this part, we present in detail the translation process from functional datatypes to meta-models. We start in Section 2.1 by giving an overview of our methodology, then we introduce the source and the target of the transformation in Sections 2.2 and 2.3 respectively. The essence of the translation is further developed in Section 2.4.

2.1 Methodology

Model Driven Engineering (MDE) is a software development methodology where the (meta-)models are the central elements in the development process. A meta-model defines the elements of a language. The instances of these elements are used to construct a model of the language. A model transformation is defined by a mapping from elements of the source meta-model to those of the target meta-model. Consequently, each model conforms to the source meta-model and can be automatically translated to an instance model of the target meta-model. The Object Management Group (OMG) [8] defined the Model Driven Architecture (MDA) standard [9], as specific incarnation of the MDE.

We apply this method in order to define a generic transformation process from datatypes (used in functional programming) to Ecore models. Figure 2 shows an overview of our approach. Using an EBNF representation of the datatype definition grammar [3], we derive a meta-model of datatypes. This meta-model is the source meta-model of our transformation. We also define a subset of the Ecore meta-model [10] to be the target meta-model. In order to perform the transformation, we defined a set of transformation rules (detailed in Section 2.4) that maps components of the meta-model of datatypes to those of Ecore Meta-model. These rules have been implemented in the application presented in Section 3.2.

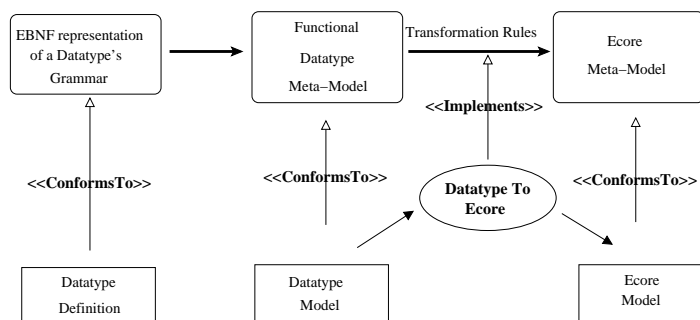


Fig. 2. Overview of the Transformation Method

2.2 Source Meta-Model : The Datatype Meta-Model

Functional programming supplies us with a rich way to describe data structures. However, since some features are not supported by *Ecore*, we have only defined a subset, that contains the essential elements composing datatypes. Figure 3 depicts the datatype metamodel that is constructed from a subset of datatype's declarations grammar [3].

A *Module* may contain several *Type Definitions*. Each *Type Definition* has a *Type Constructor*. It corresponds to the data types' name. It is also composed of at least one *Constructor Declaration*. These declarations are used to express variant types. *Type declarations* have names, it is the name of a particular type case. It takes as argument some (optional) *type expressions* which can either represent a *Primitive Type* (*int*, *bool*, *float*, etc.) or also a data type defined previously in the module. The *list* option is used to represent lists in functional programming. The *type option* feature describes the presence or the absence of a value. The *ref* option is used for references (pointers).

We enriched the type definition grammar with a new element named *Accessor*. It is a function introduced by a special annotation (**@accessor**). It allows to assign a name to a special field of the type declaration. This element is essential for the transformation process, its absence would lead to nameless structural features.

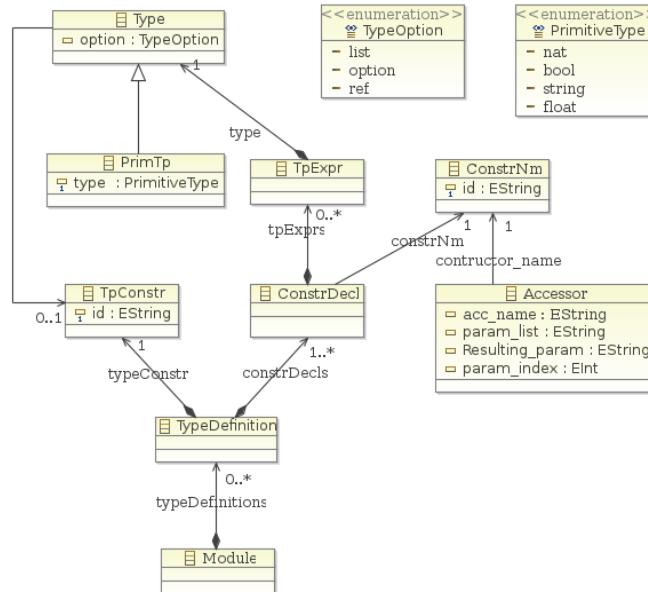


Fig. 3. Datatype Meta-model

2.3 Target Meta-Model: The Ecore Meta-Model

Our target metamodel is a subset of the Ecore metamodel. Ecore is the core language of Eclipse Modeling Framework (EMF) [11]. It allows to build Java applications based on model definitions. It unifies three technologies: Java, XML and UML. Actually, it is possible to describe a model in one of the three technologies and generate it in the other two. It also allows to develop and integrate Eclipse plug-ins.

The Meta Object Facility (MOF) standardized by the OMG defines a subset of UML class diagram [12]. It represents the Meta-Meta-Model of UML. **Ecore** is comparable to MOF but simpler. They are similar in their ability to specify classes, structural and behavioral features, inheritance and packages.

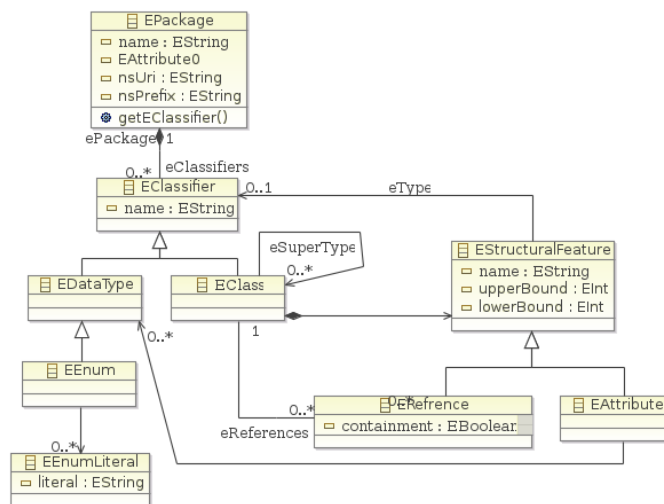


Fig. 4. Simplified subset of the Ecore Meta-model

Figure 4 represents a subset of the Ecore language. This subset contains essentially the elements that are needed for the transformation process. Its main components are:

- The **EPackage** is the root element in serialized Ecore models. It encompasses **EClasses** and **EDataTypes**.
- The **EClass** component represents classes in Ecore. It describes the structure of objects. It contains **EAttributes** and **EOperations**.
- The **EDataType** component represents the types of **EAttributes**, either pre-defined (types: Integer, Boolean, Float, etc.) or defined by the user. There is a special datatype to represent enumerated types **EEnum**, each enumeration is called **EEnumLiteral**.

- **EReferences** is comparable to the UML Association link. It defines the kinds of the objects that can be linked together. The **containment** feature is a Boolean value that makes a stronger type of relations. When it is set to true, it represents a whole/part relationship known as “by-value aggregation” in UML.

2.4 From Datatypes to Meta-Models

The transformation method is from functional datatypes to **Ecore** meta-models. To precisely define transformation rules, the transformation method is presented in a formal notation by the $Tr()$ function. In each case we start by an informal description, then we present it formally and finally we show an effective exemple.

$$Tr : DataTypes \longrightarrow Ecore\ Meta-model$$

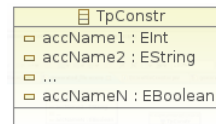
The following translation functions are given for a concrete syntax in the style of Caml [4]. Since most functional languages (including the language of proof assistants) have great similarities, the concrete syntax can be mapped to different functional languages.

Rule DatatypeToEClass When the datatype is formed of only one constructor, it is translated to an **EClass**. The **EClass** name is the name of the type constructor.

$$\begin{aligned} Tr(tpConstr = cn\ t_1 \dots t_n) = & createEClass(); \\ & setName(tpConstr); \\ & Tr_{type}(acc_i, t_i) \\ & / 1 \leq i \leq n \end{aligned}$$

Example:

```
datatype tpConstr =
  Cn of int * string * ... * bool
```

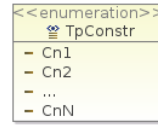


Rule DatatypeToEEnum Datatypes composed only of constructors (without *typeexprs*) are translated to **EEnums** which are usually employed to model enumerated types in **Ecore**. There, each constructor from the datatype model is translated into an **EEnumLiteral**.

$$\begin{aligned}
 Tr(tpConstr = cn_1 | \dots | cn_p) &= createEEnum(); \\
 &\quad setName(tpConstr); \\
 Tr_{constrNm}(cn_i) &= EEnumLiteral(cn_i) \quad / 1 \leq i \leq p \\
 Tr_{constrNm}(cn_i) &= EEnumLiteral(cn_i) \quad / 1 \leq i \leq p
 \end{aligned}$$

Example:

```
datatype tpConstr =
  Cn1 | Cn2 | ... | CnN
```

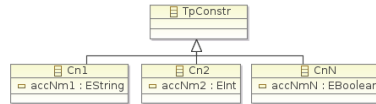


Rule DatatypeToEClasses When constructor declarations are composed of more than one constructor declaration containing type expressions: a first **EClass** is created to represent the type constructor (*tpConstr*). Then, for each constructor, an **EClass** is created too, and inherits from the *tpConstr* one.

$$\begin{aligned}
 Tr(tpConstr = cd_1 | \dots | cd_n) &= createEClass(); \\
 &\quad setName(tpConstr); \\
 &\quad Tr_{decl}(cd_i) \\
 &\quad / 1 \leq i \leq n \\
 Tr_{decl} : ConstructorDeclaration &\rightarrow EClass \\
 Tr_{decl}(cn_i \ t_1 \dots \ t_m) &= createEClass(); \\
 &\quad setName(cn_i); \\
 &\quad setSuperType(EClass(tpConstr)); \\
 &\quad Tr_{type}(acc_j, t_j) \\
 &\quad / 1 \leq j \leq m
 \end{aligned}$$

Example:

```
datatype tpConstr =
  Cn1 of string
  Cn2 of int
  ...
  CnN of bool
```

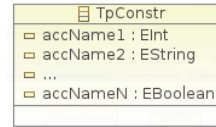


Rule PrimitivTypeToEAttribute If a type expression is formed of a primitive type, the translation function generates a new **EAttribute**. The name of this **EAttribute** is the name of its corresponding accessor, and its type is the EMF representation of the the primitive type : **EInt** for *int*, **EBoolean** for *bool*, **EString** for *string*, etc.

$$\begin{aligned} Tr_{type} : (accessor, type) &\longrightarrow EStructuralFeature \\ Tr_{type}(acc, primTp) &= createEAttribute(); \\ &\quad setName(acc); \\ &\quad setType(primTp_{EMF}); \end{aligned}$$

Example:

```
datatype tpConstr =
  Cn of int * string * ... * bool
```

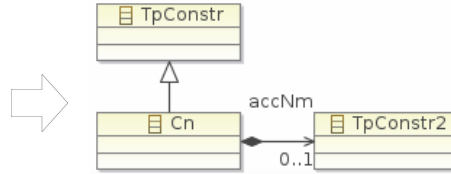


Rule TypeToEReference When a type expression contains a type which is not a primitive type, the latter has to be previously defined in the Isabelle *theory*. Then, a containment link is created between the current **EClass** and the **EClass** referenced by type constructor, and the multiplicity is set to 1.

$$\begin{aligned} Tr_{type} : (accessor, type) &\longrightarrow EStructuralFeature \\ Tr_{type}(acc, tpConstr) &= createEReference(); \\ &\quad setName(acc); \\ &\quad setType(tp_constr); \\ &\quad setContainment(true); \\ &\quad setLowerBound(1); \\ &\quad setUpperBound(1); \end{aligned}$$

Example:

```
datatype tpConstr=
  Cn of tpConstr2
```

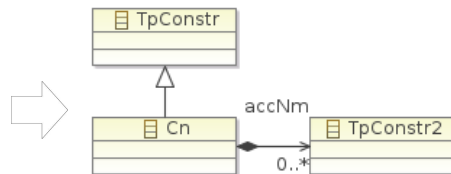


Rule TypeOptionToMultiplicity The type expressions can also appear in the form of a *type list*. In this case the multiplicity is set to $0..*$. The type expression *type option* is used to express whether a value is present or not. It returns **None**, if it is absent and **Some** value, if it is present. This is modeled by changing the cardinality to $0..1$.

$$\begin{aligned}
 Tr_{type} : (accessor, type) &\longrightarrow EStructuralFeature \\
 Tr_{type}(acc, t \text{ list}) &= Tr_{type}(acc, t) \\
 &\quad setLowerBound(0); \\
 &\quad setUpperBound(*); \\
 Tr_{type}(acc, t \text{ option}) &= Tr_{type}(acc, t) \\
 &\quad setLowerBound(0); \\
 &\quad setUpperBound(1);
 \end{aligned}$$

Example:

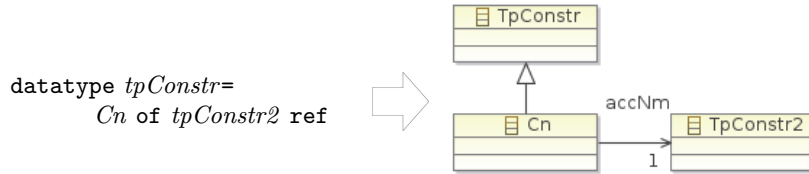
```
datatype tpConstr=
  Cn of tpConstr2 list
```



The last case that we deal with, is *type ref* which is used to represent pointers. It is translated to references without containments.

$$\begin{aligned}
 Tr_{type}(acc, t \text{ ref}) &= Tr_{type}(acc, t) \\
 &\quad setContainment(False);
 \end{aligned}$$

Example:



3 Case Study

In this section, we apply the method presented in Section 2 on a detailed example that consists of a Domain Specific Language. We start by the DSL definition, then we show the architecture of the application before finishing with the effective results of the transformation.

3.1 Presentation of the Case Study

We are currently working on a real-time dialect of the Java language allowing us to carry out specific static analyses of Java programs. We only sketch this language here; details are described in [13]. This language is not a genuine subset of Java, since we have added annotations characterizing timing behavior of program parts that are inserted in particular comments into the program. Neither is the language a superset of Java, because we have to impose syntactic restrictions on the shape of the program, and also static restrictions on the number of objects that are allocated.

All this made us opt for writing our own syntax analysis, which is integrated into the Eclipse Xtext environment [14]. After syntax analysis and verification of the above-mentioned static restrictions, the program together with its timing annotations is translated to Timed Automata (TA) for model checking. The language is currently not entirely stable and will be modified while we refine and improve the translation from Java to TA, and while the formal model evolves.

The formal aspect comes into play at the following point: We are currently developing a real-time semantics of Java in the proof assistant Isabelle, based on an execution semantics using inductive relations. Performing the translation for the whole language description would generate a huge metamodel that couldn't be presented in the paper. We thus choose to present only an excerpt of it, corresponding to a method definition.

Figure 6 shows the datatype definitions in the Isabelle proof assistant, where a method definition is composed of a method declaration, a list of variables, and statements. Each method declaration has an access modifier that specifies its kind. It also has a type, a name, and some variable declarations. The *stmt* datatype describes the statements allowed in the method body: Assignments,

Conditions, Sequence of statements, Return and the annotation statement (for timing annotations). In this example we use Booleans, integers, strings for types and values.

3.2 Implementation: DatatypesToEcore

Our approach is implemented using the Eclipse environment which includes among others

- Eclipse Modeling Framework (EMF) [11]: a framework for modeling and code generation that builds tools and applications based on data models.
- Eclipse Modeling Project (EMP) [10]: a framework allowing the manipulation of DSLs by defining their (textual/graphical) concrete syntax based on a corresponding metamodel.

Figure 5 shows the architecture of our application. There, green arrows represent model transformations or code generation. The base element is an Isabelle *theory* where both of the datatypes and the properties to be checked are defined. The corresponding meta-model is generated using the translation function described in Section 2.4. Starting from a generated Ecore meta-model, we use the Xtext tool to define a textual concrete syntax. First, Xtext builds an EBNF grammar depending on the structure of the metamodel. The grammar is then adapted using the right key words of the language, yielding a textual editor as an Eclipse plug-in.

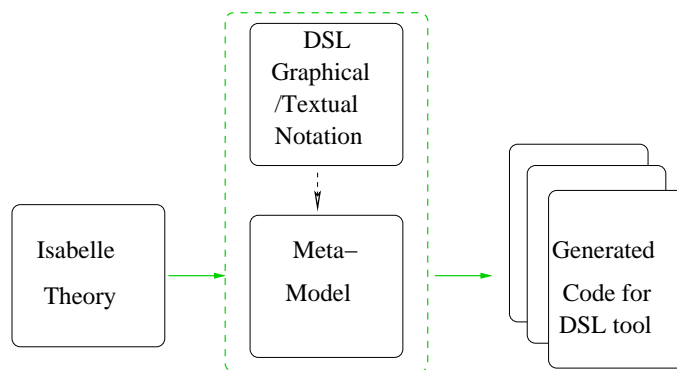


Fig. 5. Datatype To Ecore implementation architecture

3.3 Applying the Transformation

Figure 6 shows a datatype taken from the Isabelle *theory* where the verifications were performed. Due to lack of space we do not present them in the paper.

This part of the *theory* was given as input to the implementation of our translation rules presented in Section 2.4. The resulting **Ecore** diagram is presented in Figure 7.

As it is shown on the figure, data type definitions built only of type constructors (*Tp*, *AccModifier*, *Binop*, *Binding*) are treated as enumerations in the metamodel. Whereas *Datatype MethodDecl* composed of only one constructor derive a single class. As for type expressions that represent list of types (like *accModifier list* in *varDecl*), they generate a structural feature in the corresponding class and their multiplicities are set to $(0..*)$. The result of type definitions containing more than one constructor and at least a type expression (*stmt* and *expr*) is modeled as a number of classes inheriting from a main one. Finally, the translation of the *int*, *bool* and *string* types is straightforward. They are translated to respectively **EInt**, **EBoolean** and **EString**.

```

datatype binop = BArith| BCompar| BLogic
datatype value = BoolV bool
                |IntV int
                |StringV string
                |VoidV
datatype binding = Local| Global
datatype var = Var binding string
datatype expr = Const value
                |VarE var
                |BinOperation binop expr expr
datatype tp = BoolT| IntT| VoidT| StringT
datatype stmt = Assign var expr
                |Seq stmt stmt
                |Cond expr stmt stmt
                |Return expr
                |AnnotStmt int stmt

datatype accModifier =
    Public |Private |Abstract|Static |Protected |Synchronized
datatype varDecl =
    VarDecl (accModifier list) tp int
datatype methodDecl =
    MethodDecl (accModifier list) tp string (varDecl list)
datatype methodDefn =
    MethodDefn methodDecl (varDecl list) stmt

```

Fig. 6. Datatypes in Isabelle

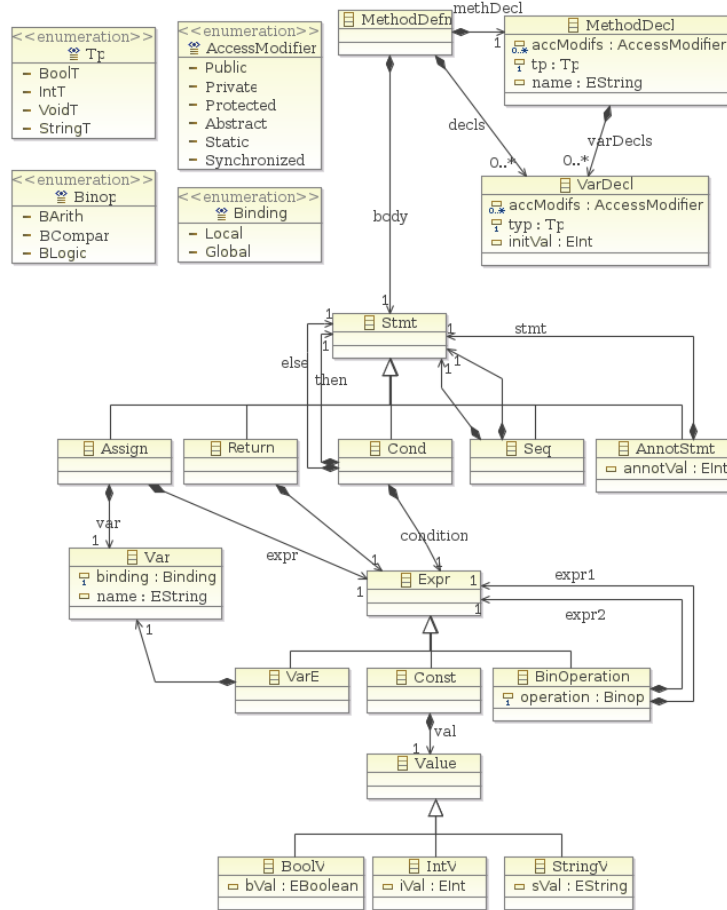


Fig. 7. Resulting Ecore Diagram after Transformation

4 Related Work

EMF models are comparable to Unified Modeling Language Class diagrams. For this fact we are interested in the mappings from other formal languages to UML Class diagrams. Some research is dedicated to establishing the link between these two formalisms. We cite the work of *Idani & al.* that consists of a generic transformation of UML models to B constructs [15] and vice-versa [16]. The authors propose a metamodel-based transformation method based on defining a set of structural and semantic mappings from UML to B (a formal method that allows to construct a program by successive refinement, using abstract specifications).

Similarly, there is an MDE based transformation approach for generating Alloy (a textual modeling language based on first order logic) specifications from UML class diagrams and backwards [17], [18].

Delahaye & al. describe in [19] a formal and sound framework for transforming Focal specification into UML models.

These methods enable to generate UML component from a formal description but their formal representation is significantly different from our needs: functional data structures.

Also, graph transformation tools [20, 21] permit to define source and target metamodels all along with a set of transformation rules and use graphical representations of instance models to ease the transformation process. However, the verification functionality they offer is often limited to syntactic aspects (such as confluence of transformation rules) and does not allow to model deeper semantic properties (such as an operational semantics of a programming language and proofs by bisimulation).

Our approach combines the two views by offering the possibility to define the abstract syntax of a DSL, to run some verifications on the top of it and to generate the corresponding metamodel to graphically document the formal developments. Furthermore, this metamodel can be used to easily generate a textual editor using Xtext facilities.

5 Conclusion

Our work constitutes a first step towards a combination of interactive proof and Model Driven Engineering. We have presented a generic method based on MDE for transforming data type definitions used in proof assistants to Class diagrams.

The approach is illustrated with the help of a Domain Specific Language developed by ourselves. It is a Java-like language enriched with annotations. Starting from data type definitions, set up for the semantic modeling of the DSL we have been able to generate an EMF meta-model. In addition to its benefits for documenting and visualizing the DSL, it is manipulated in the Eclipse workbench to generate a textual editor as an Eclipse plug-in.

Currently, we are working on extending subset of data type definitions by adding a way to transform parameterized types to generic types in Ecore. And coupling our work with the generation of provably correct object oriented code from proof assistants. Moreover, we intend to work on the opposite side of transformation, the possibility to generate data structure definitions from class diagrams.

References

1. van Deursen, A., Klint, P., Visser, J.: Domain-specific languages: An annotated bibliography. *SIGPLAN Notices* **35** (2000) 26–36
2. <http://coq.inria.fr/>: Coq proof assistant website (2012)

3. Nipkow, T., Paulson, L., Wenzel, M.: Isabelle/HOL. A Proof Assistant for Higher-Order Logic. Volume 2283 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2002)
4. <http://caml.inria.fr>: Caml programming language website (2012)
5. Martin Odersky et al.: An Overview of the Scala Programming Language. Technical report, EPFL (2007)
6. Bézivin, J.: Model driven engineering: An emerging technical space. In Lämmel, R., Saraiva, J., Visser, J., eds.: Generative and Transformational Techniques in Software Engineering. Volume 4143 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2006) 36–64
7. Selic, B.: The pragmatics of model-driven development. *IEEE Software* **20** (2003) 19–25
8. OMG: Meta Object Facility (MOF) Core v. 2.0 Document. (2006)
9. Kleppe, A.G., Warmer, J., Bast, W.: MDA Explained : The Model Driven Architecture : Practice and Promise. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2003)
10. Gronback, R.C.: Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit. Addison-Wesley, Upper Saddle River, NJ (2009)
11. Budinsky, F., Brodsky, S.A., Merks, E.: Eclipse Modeling Framework. Pearson Education (2003)
12. France, R.B., Evans, A., Lano, K., Rumpe, B.: The UML as a formal modeling notation. *Computer Standards & Interfaces* **19** (1998) 325–334
13. Baklanova, N., Strecker, M., Féraud, L.: Resource Sharing Conflicts Checking in Multithreaded Java Programs. Informal Proceedings FAC’12 (2012)
14. Eclipse Community: Tutorials and documentation for Xtext 2.0 (2011) <http://www.eclipse.org/Xtext/documentation/>.
15. Idani, A., Boulanger, J.L., Philippe, L.: A generic process and its tool support towards combining UML and B for safety critical systems. In Hu, G., ed.: CAINE, ISCA (2007) 185–192
16. Idani, A.: UML models engineering from static and dynamic aspects of formal specifications. In Halpin, T.A., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Soffer, P., Ukor, R., eds.: BMMDS/EMMSAD. Volume 29 of Lecture Notes in Business Information Processing., Springer (2009) 237–250
17. Shah, S.M.A., Anastasakis, K., Bordbar, B.: From UML to Alloy and back again. In Ghosh, S., ed.: MoDELS Workshops. Volume 6002 of Lecture Notes in Computer Science., Springer (2009) 158–171
18. Anastasakis, K., Bordbar, B., Georg, G., Ray, I.: UML2Alloy: A challenging model transformation. In Engels, G., Opdyke, B., Schmidt, D.C., Weil, F., eds.: MoDELS. Volume 4735 of Lecture Notes in Computer Science., Springer (2007) 436–450
19. Delahaye, D., Étienne, J.F., Viguié Donzeau-Gouge, V.: A Formal and Sound Transformation from Focal to UML: An Application to Airport Security Regulations. In: UML and Formal Methods (UML&FM). Innovations in Systems and Software Engineering (ISSE) NASA Journal, Kitakyushu-City (Japan), Springer (2008)
20. de Lara, J., Vangheluwe, H.: Using AToM³ as a meta-case tool. In: Proceedings of the 4th International Conference on Enterprise Information Systems (ICEIS), Ciudad Real, Spain (2002) 642–649
21. Ehrig, K., Ermel, C., Hänsgen, S., Taentzer, G.: Generation of visual editors as Eclipse plug-ins. In: Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering. ASE ’05, New York, NY, USA, ACM (2005) 134–143

Intelligent Testing in Kyiv: Analytical and Deductive Paradigms and their Implementation

Vitaly Klimenko¹ and Alexander Lyaletski²

¹ Institute for Problems of Mathematical Machines and Systems
Glushkov avenue, 42, 03187 Kyiv, Ukraine
klimenko@imm-sp.kiev.ua

² Institute Faculty of Cybernetics, Kyiv National Taras Shevchenko University
Glushkov avenue, 4D, 03680 Kyiv, Ukraine
lav@unicyb.kiev.ua

Abstract. We describe an experience of the Kiev schools on analytical transformations and automated deduction with respect to intelligent testing of knowledge obtained through distant learning with or without the use of electronic textbooks. Leaving aside the simple "choose from prescribed answers" tests, intelligent testing of two kinds is considered: one of them relies on symbolic computation and the other on deduction.

Keywords: Analytical transformations, automated deduction, electronic learning, knowledge testing, intelligent testing, on symbolic computation and the other on deduction.

Key Terms. Development, Machine Intelligence, Software Engineering Process

1 Introduction

At the beginning of the 1960th, Academician V.M.Glushkov initiated two ways of the development of computer-aided mathematics in Kiev: one was concerned with symbolic computations (i.e. computer algebra systems in the modern terminology) and the other with the automatization of deduction steps (i.e. automated reasoning systems). Now, these two ways play important role in information technologies. That is why it is interesting to know impact they can have on the development of intelligent testing in e-learning in current days.

There exist a great number of software systems for authoring of "electronic textbooks" for various disciplines being taught in secondary schools, colleges, and universities. Their common feature is their orientation towards a broad spectrum of educational branches and, owing to this, towards the simplest kind of examination of trainee's knowledge based on choosing a right answer from a number of alternatives

proposed by an examiner. The downside of this technique is that it gives the trainee an incentive to guess a right answer rather than really look for it, which does not allow a tutor to estimate trainee knowledge correctly. A list of "prescribed answers" is notably inconvenient for mathematical disciplines, where a solution to a problem often consists in deriving an analytic (symbolic) expression or in a formal proving when a chain of deductive steps assuring the validity of a statement under consideration must be constructed. Thus, we have the following ways for the computer-aided testing of knowledge obtained by a trainee in the (e-)learning of a subject: the query-answering method, the analytical transformation, the deductive construction, and, perhaps, their combinations.

The state of the art in symbolic computation and automated reasoning has initiated transition from the simple "choose-an-answer" testing to the more intelligent and complex ones: the analytical and deductive reasoning. The first approach is suitable for testing on the base of finding analytical solutions of tasks in algebra, trigonometry, physics, and so on (for both secondary schools and higher education institutions). The second one is useful in studying a mathematical theory allowing its complete formalization for computer checking of logical steps of a trainee proving a certain sentence of a theory under consideration (the same concerns any knowledge domain, where formalization and deduction are admissible).

The present paper is devoted to the brief description of some of the approaches of Ukrainian researchers relating to symbolic (analytical) computation and automated reasoning that can be used for the construction of software tools for intelligent testing (other than the "choose-an-answer" method).

2 Symbolic Computation

Testing on the base of symbolic computation is needed when a solution for an equation must have the form of an analytical (symbolic) expression, for example, a root of an algebraic equation or an equation in partial derivatives. In order to perform such a testing, we need a "shell" that can assure that the symbolic expression proposed by the examinee is correct, that is, it can be transformed into a formula, given by an examiner by means of symbolic computation. Such analytical verification is very appropriate for testing in various domains of physics, trigonometry, algebra, and so on. In the first place, it requires the following generic tools:

- tools for performing arbitrary-precision computation for integers, rationals and complex numbers;
- methods for determining whether two symbolic expressions are equal; these are usually based on various systems of term rewriting rules;
- methods for term normalization (in particular, normalization to certain conventional mathematical forms);
- tools for analytical transformations of mathematical expressions defined in the terms of hierarchical data structures of arbitrary complexity.

The Institute for Problems of Mathematical Machines and Systems National, Academy of Sciences of Ukraine, (IPMMS NASU) started research in this domain in 1960s and created a family of hierarchically developing computer algebra systems in the frame of the "Analytic" project under the leadership of Academician

V.M.Glushkov, which was strongly oriented to assisting in performing analytical transformation operations such as algebraic simplification, mathematical derivation and integration, etc. The specialized computer series "MIR" (Mashina dlya Inzhenernykh Raschetov - Engineering Computation Machine): "MIR-1", "MIR-2", and "MIR-3" having their input languages of the three first version of the "Analitic" language [1]. Later, the developed algorithms were implemented into the SM 1410 computers ("Analitic-79" [2]) and into the usual IBM PC ("Analitic-93" [3] and "Analitic-2000" [4]). Now, the modern project versions called "Analitic-2007" [5] and "Analitic-2010" [6] are in progress and usage. Let us give a brief description of some of their features and implementation.

2.4 Analitic-2007

The "Analitic-2007" version was implemented at the beginning of 2007. It inherited all the best features of all its predecessors and differed from the previous versions by deeper algebraic transformations, more detailed classification of algebraic tools, sophisticated facilities of calculations control, and improved interactive methods.

The "Analitic-2007" programming system is intended for IBM PCs and is operated as an application for the operating system Windows-2000 and higher versions. It consists of the system kernel and a number of program packs. The compact kernel provides a user with a large quantity of programs, supports the semantic integrity of the "Analitic" language, the universality of its functional properties, and the operability of the "Analitic-2007" system in the environment of the different Windows operating systems. It performs compiling and recompiling programs and data, executing programs, transforming language objects (including programs being considered as objects of the language).

The system automatically determines the size of memory accessible for performing a program and occupies the maximal scope of accessible memory by default. A user has a possibility for determining the size of memory necessary for the normal execution of his programs. In the case of exceeding the existent memory size, the "Analitic-2007" programming system uses a virtual memory.

2.5 Analitic-2010

The last version "Analitic-2010" has significant changes in the kernel of "Analitic-2007" and it is partially transferred to the .NET platform [6]. A new user-friendly graphic interface missing in the previous versions was constructed.

When implementing "Analitic-2010", the main efforts were directed to the improvement of the operating stability of the kernel. For this, the parser was recoded without any changes in the language "Analitic". As a result, the software of the previous versions was transferred to the new one.

The new interface is oriented to the efficient handling of data in the interactive mode and the faster generation of new programs. It is equipped with a complete code editor supporting intelligent input and all the possibilities inherent in a modern integrated development environment.

Finally, we note that all the "Analytic" family systems are used actively for finding analytical solutions of tasks in mechanics, astronomy, differential equations theory. Besides, a number of experiments were performed in automated performing of analytical transformation in various fields of mathematical learning and testing, for example, in checking of algebraic and trigonometric identities. Thus, they can be very useful in e-learning.

3 Deductive Computation

Deductive testing consists in logically verifying a chain of reasoning steps expressed in a formal language. The Ukrainian approach is based on a so-called evidential paradigm [7] being the modern vision of the Evidence Algorithm advanced by V.M.Glushkov in [8]. In accordance with it, the language should be close to a natural language and it should be used in a course of training. Deductive testing is to be used in mathematical disciplines having the form of formal axiomatic theories containing logical inference rules. It may also be successfully applied in jurisprudence, where the testing consists in performing legally and logically valid reasoning steps, or in creating legal documents consistent with the current legislation.

3.1 Evidential Paradigm

The evidential paradigm [7] itself is based on the declarative way of representation and logical processing of knowledge having the form of formalized texts (containing axioms, definitions, propositions and so on, when we deal with mathematical problems). Systems exploiting this paradigm are called automated reasoning systems or, in particular, systems for automated theorem proving. Note that this approach turns out to be the most adequate for the automated logical inference search as well as for verification of a formal text (mathematical or not), namely, for checking the validity of all the reasoning steps in it.

For the purposes of deductive testing in the frame of the evidential paradigm, we adhere to the following requirements for a testing environment:

- For presentation of reasoning, a trainee must use a (semi-)formal language which is close to the natural language of mathematical publications. This language preserves the structure of the problem in question and the texts in this language can be translated into some representation convenient for computer processing.
- Each reasoning step (in a natural form) from the text under verification must be "obvious" to a computer in the sense that it can be checked by it. A checking procedure must evolve for incrementing reasoning steps as much as possible. It must combine general methods of logical inference search with heuristic reasoning techniques such as induction, case reasoning, definition handling, and so on. Such collection of reasoning techniques must also grow and evolve.
- In the case of necessity, symbolic computation methods must be attracted to the testing of trainee knowledge;
- Formal knowledge accumulated in the system (and used in training) must be organized in a hierarchical information environment.

From 1998, the evidential paradigm is actively investigating in Ukraine, mainly, at the Faculty of Cybernetics of the Kiev National Taras Shevchenko University.

3.2 SAD System

As a result of the development of the evidential paradigm, the System for Automated Deduction (SAD) has been constructed. Its basic purpose is to assist to a man in "doing" mathematics, for example, in mathematical text verifying and/or theorem proving. It can be downloaded or accessed online on the Evidence Algorithm project site "<http://nevidal.org>" (see also [9] containing the history of the development of the Evidence Algorithm and SAD system).

In accordance with the evidential paradigm, the SAD system can perform the following chain of text transformations oriented to theorem proving/verifying [10]:

- At the first level, an input text written at the original formal language ForTheL [11] being close to the natural English language of mathematical publications is syntactically analyzed. After this, it can be translated into a first-order language, which permits to perform computer inference search in different logics;
- At the second level, the goal statements are processed one-by-one by a foreground reasoner of SAD. In the verification mode, this module is intended to reduce a given proof task to a number of subtasks for a prover. At present, its toolkit contains some simplification methods on propositional level such as decomposition of a problem under consideration, reasoning by general induction, and others;
- Inference search tasks are resolved at the third level with the help of a first-order prover, for example, with the native prover Moses [12] that is based on a special goal-driven sequent calculus for classical first-order logic with equality.

The original notion of admissible substitution used in the calculus permits to preserve the initial signature of the task so that special accumulated equations can be sent to a specialized solver, e.g. an external computer algebra system. In particular, some of the tools of the "Analytic" programming systems can be used for this in the case of the existence of an according interface.

Note that the SAD system was implemented in such a way that SAD can be connected with an external first-order prover such as Otter [13], SPASS [14], or Vampire [15], E prover [16], and Prover 9 [17].

Therefore, the SAD is a good candidate for deductive testing in e-learning.

4 Conclusion

The above-given analysis of Kiev approaches to symbolic computation and deductive reasoning demonstrates that the advances made by researches at IPMMS and Kiev National Taras Shevchenko University allow introducing and implementing various forms of distant e-learning based on a more thorough and unbiased evaluation of an examinee, which can improve the quality of learning for disciplines which admit (at least partial) formalization. The next step consists in integration of analytical and deductive testing in a common framework, allowing these two forms of intelligent testing to complement and enforce each other.

When the described methods and tools for intelligent testing will be implemented and approved in practice, the next milestone can be proceed. First, these tools can be incorporated into the existing e-learning systems (taking into account the specifics of a domain under study). Second, one can use the proposed framework to design and implement entire electronic courses and textbooks, containing learning material as well as exercises for simple and intellectual testing for objective evaluation of student's knowledge.

References

1. Glushkov V.M., Bodnarchuk V.G., Grinchenko T.A., Dorodnizyna A.A., Klimenko V.P., Letichevsky A.A., Pogrebinsky S.B., Stogniy A.A., and Fishman Yu.S.: ANALITIK (an algorithmic language for description of computational processes using analytical transformations) (in Russian). *Cybernetics* 3, pp.102--134 (1971).
2. Glushkov V.M., Grinchenko T.A., Dorodnizyna A.A., Drakh A.M., Klimenko V.P., Pogrebinsky S.B., Savchak O.N., Fishman Yu.S., and Tsaryuk N.P.: ANALITIK-79 (in Russian). Technical report, Institute of Cybernetics, Kiev, USSR (1979).
3. Morozov A.A., Klimenko V.P., Fishman Yu.S., Bublik B.A., Gorovoy V.D., and Kalina E.A.: ANALITIK-93 (in Russian), *Mathematical Machines and Systems* 5, pp. 127--157 (1995).
4. Morozov A. A., Klimenko V. P., Fishman Yu. S., Lyakhov A. L., Kondrashov S. V., and Shvalyuk T. N.: ANALITIK-2000 (in Russian). *Mathematical Machines and Systems* 1,2, pp. 66--99 (2001).
5. Morozov A.A., Klimenko V.P., Fishman Yu.S., and Shvalyuk T.N.: ANALITIK-2007 (in Russian). *Mathematical Machines and Systems* 3,4, pp. 8--52 (2007).
6. Klimenko V.P., Lyakhov A.L., Gvozdik D.N., Zakharov S.A., and Shvalyuk T.N.: On the implementation of a new version of the Analytic family CAS (in Russian). In: International conference CMSEE-2010, Poltava (2010).
7. Lyaletski A. and Morokhovets M.: Evidential paradigm: a current state. In: Programme of the International Conference "Mathematical Challenges of the 21st Century", University of California, Los Angeles, USA (2000).
8. Glushkov V.M.: Some problems of automata theory and artificial intelligence (in Russian). *Cybernetics* 2, pp. 3--13 (1970).
9. Lyaletski A. and Verchinine K.: Evidence Algorithm and System for Automated Deduction: A retrospective view. LNCS, vol. 6167, pp. 411--426 (2010).
10. Anisimov A. V. and Lyaletski A. V.: The SAD system in three dimensions, In: International symposium SYNASC'06, Timisoara, Romania, pp. 85--88 (2006).
11. Vershinin K. and Paskevich A.: ForTheL – the language of formal theories. *International Journal of Information Theories and Applications*, vol. 7, no. 3, pp. 120--126 (2000).
12. Lyaletski A., Paskevich A., and Verchinine K.: Theorem proving and proof verification in the system SAD. LNCS, vol. 3119, pp. 236--250 (2004).
13. The Otter automated deduction system, <http://www.mcs.anl.gov/research/projects/AR/otter/>.
14. The SPASS Prover, <http://www.spass-prover.org/>
15. The Vampire prover, <http://www.vprover.org/>
16. The E Theorem Prover, <http://www4.informatik.tu-muenchen.de/~schulz/E/E.html>
17. The Prover9 prover, www.cs.unm.edu/~mccune/prover9/

Semantics-based Logics over Hierarchical Nominative Data

M(N).S. Nikitchenko¹ and S.S. Shkilniak¹

¹ Department of Theory and Technology of Programming
Taras Shevchenko National University of Kyiv
01601, Kyiv, Volodymyrska st, 60
Tel.: +38044 2590519
nikitchenko@unicyb.kiev.ua

Abstract. In the paper new logics oriented on hierarchical data are developed. Algebras of partial predicates over such data with special compositions as operations form a semantic base for constructed logics. Characteristic property of such logics is the usage of composite names in their languages. Semantic properties of these logics are studied; corresponding sequent calculi are defined, their soundness and completeness are proved for logics of renominative level.

Keywords: partial predicates, program semantics, nominative data, composition, logic, soundness, completeness.

Key Terms. Research, MathematicalModel, FormalMethods, MachineIntelligence

1 Introduction

Mathematical logic is widely used in formal program development, analysis, and verification. Still, some discrepancies can be admitted between traditional logic and problems to be solved. For example:

- semantics of programs is presented by partial functions, whereas in traditional logic total functions and predicates are usually considered;
- programming languages have a developed system of data types, whereas traditional logic prefers to operate with simple unstructured types (sorts);
- semantic aspects of programs prevail over syntactical aspects, whereas in traditional logic we have the inverse situation.

Discrepancies mentioned above complicate the usage of logic for program development and verification. In this paper we propose to take program models as an initial point and construct logics semantically based on such models.

To realize this idea we should first construct adequate models of programs. To tackle this problem we use composition-nominative approach to program formalization [1], which aims to construct a hierarchy of program models of various abstraction levels and generality. The main principles of the approach are the following.

- *Development principle* (from abstract to concrete): program notions should be introduced as a process of their development that starts from abstract understanding capturing essential program properties and proceeds to more concrete considerations.
- *Principle of integrity of intensional and extensional aspects*: program notions should be presented in the integrity of their intensional and extensional aspects. The intensional aspects in this integrity play a leading role.
- *Principle of priority of semantics over syntax*: program semantic and syntactical aspects should be first studied separately, then in their integrity in which semantic aspects prevail over syntactical ones.
- *Compositionality principle*: programs can be constructed from simpler programs (functions) with the help of special operations, called compositions, which form a kernel of program semantics structures.
- *Nominativity principle*: nominative (naming) relations are basic ones in constructing data and programs.

Here we have presented only principles relevant to the topic of the article; richer system of principles is developed in [2]. The above principles specify program models as *composition-nominative systems* (CNS) [1]. Such a system may be considered as a triple of simpler systems: composition, description, and denotation systems. A composition system defines semantic aspects of programs, a description system defines program descriptions (syntactical aspects), and a denotation system specifies meanings (referents) of descriptions. We consider semantics of programs as partial functions over class of data processed by programs; compositions are n -ary operations over functions. Thus, composition system can be specified as two algebras: data algebra and functional algebra.

Functional algebra is the main semantic notion in program formalization. Terms of this algebra define syntax of programs (descriptive system), and ordinary procedure of term interpretation gives a denotation system.

CNS can be used to construct formal models of various programming, specification, and database languages [1–4]. The program models presented by CNS are mathematically simple, but specify program semantics rather adequately; program models are highly parametric and can in a natural way represent programs of various abstraction levels; there is a possibility to introduce on a base of CNS the notion of special (abstract) computability and various axiomatic formalisms [5–7].

CNS are classified in accordance with levels of abstraction of their parameters: data, functions, and compositions. In this article levels of program models are induced by abstraction levels of data.

Data are considered at three levels: abstract, Boolean, and nominative. At the abstract level data are treated as "black boxes", thus no information can be extracted. At the Boolean level to abstract data new data considered as "white boxes" are added. Usually, these are logical values T (true) and F (false) from the set $Bool$. At the nominative level data are considered as "grey boxes", constructed of "black" and "white boxes" with the help of naming relations. The last level is the most interesting for programming. Data of this level are called *nominative data*. The class of *nominative data* over a set of names V and class of basic values W can be defined inductively or as the least fixed point of the recursive definition

$ND(V, W) = W \cup (V \xrightarrow{m} ND(V, W))$, where $V \xrightarrow{m} ND(V, W)$ is the class of partial multi-valued (non-deterministic) functions.

To present nominative data we use the form $d = [v_i \mapsto a_i \mid i \in I]$. *Nominative membership relation* is denoted by \in . Thus, $v_i \mapsto a_i \in d$ means that the value of v_i in d is defined and is equal to a_i .

The class $ND(V, W) \setminus W$ is called the class of *proper nominative data*, or *hierarchical nominative data*; data from the class $V \xrightarrow{m} W$ will be called *flat nominative data*.

Concretizations of nominative data can represent various data structures, such as records, arrays, lists, relations, etc. [1, 4]. For example, a set $\{s_1, s_2, \dots, s_n\}$ can be presented as nominative data $[1 \mapsto s_1, 1 \mapsto s_2, \dots, 1 \mapsto s_n]$, where 1 is treated as a standard name. Thus, we can formulate the following *data representation principle*: program data can be presented as concretizations of nominative data.

The levels of data abstraction formulated above may be treated as data intensionals. They respectively specify three levels of semantics-based program models: abstract, Boolean, nominative. The models of each level constitute extensionals of that level intensional. Program models of abstract level are very poor (actually, only sequencing compositions can be defined). Program models of Boolean level are richer and permit to define structured programming constructs (sequence, selection, and repetition). This level is still too abstract and does not explicitly specify data variables. At last, models of nominative level permit to formalise compositions of traditional programming. This level (its intensional) involves variables of different types. Consider, for example, a simple educational programming language WHILE [8], which is based on three main syntactical components: arithmetic expressions, Boolean expression, and statements. States of WHILE programs are considered as partial functions from the set V of variables to the set Z of values and here are denoted by ${}^V Z (= V \rightarrow Z)$. Thus, semantics of these components is the following: arithmetic expressions specify functions of the type ${}^V Z \rightarrow Z$ (we call them partial quasiary functions), Boolean expressions define functions of the type ${}^V Z \rightarrow Bool$ (partial quasiary predicates), statements specify functions of the type ${}^V Z \rightarrow {}^V Z$ (partial bi-quasiary functions). Note that in our terminology ${}^V Z$ is a class of single-valued flat nominative data.

Example 1. Consider a Boolean expression $x < y$. Its semantics is formalized as a partial quasiary predicate $less : {}^V Z \rightarrow Bool$. This predicate is undefined on flat nominative data $[x \mapsto 5, u \mapsto 4]$ (we write $less([x \mapsto 5, u \mapsto 4]) \uparrow$), is defined on $[x \mapsto 5, u \mapsto 4, y \mapsto 2]$ with value F (we write $less([x \mapsto 5, u \mapsto 4, y \mapsto 2]) \downarrow = F$). Note that if a value of $less$ is defined on some data, then the predicate is defined with the same value on any extension of this data. Thus, $less([x \mapsto 5, u \mapsto 4, y \mapsto 2, v \mapsto 4]) \downarrow = F$, $x, u, y, v \in V$. This property is called *equitonicity* (a special case of monotonicity). A specific new composition is renomination $R_{x_1, \dots, x_n}^{v_1, \dots, v_n}$,

$$\text{e.g. } (R_{y, v}^{x, y} (less))([x \mapsto 5, u \mapsto 4, y \mapsto 2, v \mapsto 4]) = less([x \mapsto 2, u \mapsto 4, y \mapsto 4, v \mapsto 4]) = T.$$

More elaborated programming languages work with hierarchical nominative data. In such languages composite names like $x_1.x_2. \dots .x_n$ are used to access data components. The details can be found in [2].

Having described program models of various abstraction levels, we can now start developing semantics-based logics which correspond to such models. Such logics will be called *composition-nominative logics* (CNL). Analysis of constructed program models shows that the main semantic notion of mathematical logic – the notion of predicate – can be defined at the Boolean level. At this level predicates are considered as partial functions from a class of abstract data A (with abstract intensional) to $Bool$. In this case such compositions as disjunction \vee , negation \neg , etc, can be defined. These compositions are derived from Kleene's strong connectives [9]. Thus, the main semantic objects are algebras of partial predicates of the type $\langle A \rightarrow Bool; \vee, \neg \rangle$. The obtained logics may be called propositional logics of partial predicates. Such logics are rather abstract, therefore their further development is required at the nominative level. As was mentioned earlier, at this level we have two sublevels determined respectively by flat and hierarchical nominative data.

Three kinds of logics can be constructed from program models at the flat nominative data level:

- logics, which use only partial quasiary predicates (pure predicate logic);
- logics, which use additionally partial quasiary functions (predicate-function logics);
- logics, which use also bi-quasiary functions (program logics).

The first type of logics will generalize classical pure predicate logics, the second type – classical predicate logic (with functions and equality), and the third type can present various logics, which use program constructs.

Here we give a short characteristic only to composition-nominative pure predicate logics; predicate-function logics are described in [3]; as to composition-nominative program logics some initial variants are presented in [3, 7].

From semantic point of view the main distinction of CNL from classical first-order logics is usage of partial quasiary predicates instead of total n -ary predicates; this leads to algebras of quasiary predicates with compositions as operations. From syntactical point of view formulas of CNL are simply terms of algebras of quasiary predicates.

The main compositions that can be additionally specified at the nominative level are renomination $R_{x_1, \dots, x_n}^{v_1, \dots, v_n}$ (denoted also $R_{\bar{x}}$) and quantification $\exists x$. These compositions use subject names as parameters. CNL of renominative level are based on algebras of the type $\langle {}^V A \rightarrow Bool; \vee, \neg, R_{\bar{x}} \rangle$, CNL of quantifier level – $\langle {}^V A \rightarrow Bool; \vee, \neg, R_{\bar{x}}, \exists x \rangle$. Properties of these algebras determine calculi for corresponding logics.

Note, that renomination (primarily in syntactical aspects) is widely used in classical logic, lambda-calculus, and specification languages like Z-notation [10], B [11], TLA [12], etc. Here we will give explicit semantic definition of this operation (cf. with [13]).

To preserve properties of classical first-order logic we should restrict the class ${}^V A \rightarrow Bool$ of quasiary predicates. Namely, we introduce a class of equitone predicates and its different variations such as maxitotal equitone, local-equitone, equicompatible, and local-equicompatible classes [3]. Logics based on equitone and maxitotal equitone predicates are the “closest” generalization of classical first-order logic that preserve its main properties. These logics are called *neoclassical logics* [3].

The current article continues investigations of pure predicate logics over hierarchical nominative data initiated in [14]. Here we prove soundness (correctness) and completeness of the constructed logics. The distinctive feature of such logics is the usage of composite names of the form $x_1.x_2. \dots .x_n$ as parameters of renomination and quantification compositions.

The article is structured as follows: the first section is introduction, in the second section operations over hierarchical data are introduced and their properties are studied, the third section is devoted to compositions over predicates. In the fourth section semantic models and corresponding languages of logics are described, and the fifth section is devoted to definition of sequent calculi for some of the described logics.

Notions not defined here we interpret in sense of [3].

2 Hierarchical Nominative Data

Class of hierarchical nominative data $ND(V, A)$ over classes of basic names V and basic values A is defined inductively:

- 1) $ND_0(V, A) = A$ – nominative data of rank 0;
- 2) $ND_{k+1}(V, A) = A \cup (V \xrightarrow{n} ND_k(V, A))$ – nominative data of rank less or equal to $k+1$.

Then $ND(V, A) = \bigcup_{k \geq 0} (V \xrightarrow{n} ND_k(V, A))$.

Here $V \xrightarrow{n} ND_k(V, A)$ is the set of all finite single-valued mappings from V to $ND_k(V, A)$. Note, that we restrict nominative data to be single-valued mappings. This guaranties unambiguity of naming for data components. An empty nominative data has rank 0.

The set of *hierarchical nominative data* is defined as follows: $HD(V, A) = ND(V, A) \setminus A$.

The value of name u in data d is equal to $d(u)$, but we also write $u:d$ in style of denaming operation. For a composite name $u = y_1.y_2. \dots .y_n$ notation $u:d$ means $y_n:(\dots(y_2:(y_1:d))\dots)$. We drop a component $x \mapsto u:\delta$, if $u:\delta$ is undefined.

Hierarchical data can be represented also as oriented trees with edges labeled by basic names and leafs labeled by basic values.

Any hierarchical data d can be represented as a flat nominative data with composite names – elements of the set V^+ . These composite names are non-empty words in the alphabet V formed by concatenation “.” of basic names along the path from the root to leafs in the tree representing d .

Example 2. Let $[x \mapsto [y \mapsto 1, z \mapsto 2], y \mapsto [x \mapsto 3, y \mapsto [x \mapsto 0, y \mapsto 0, z \mapsto 1]]], z \mapsto 2, u \mapsto [x \mapsto [x \mapsto 0, u \mapsto 1], z \mapsto 3]]$ be hierarchical data. Its flat representation is

$[x.y \mapsto 1, x.z \mapsto 2, y.x \mapsto 3, y.y.x \mapsto 0, y.y.y \mapsto 0, y.y.z \mapsto 1, z \mapsto 2, u.x.x \mapsto 0, u.x.u \mapsto 1, u.z \mapsto 3]$.

Such representations are called *flat normal forms (FNF)* of hierarchical data. Due to the unambiguity of naming all (composite) names of FNF must be different;

moreover, they should be *incomparable*. Now it is possible to write $[x.y \mapsto \alpha, x.u \mapsto \beta, \dots]$ in place of $[x \mapsto [y \mapsto \alpha, u \mapsto \beta, \dots]]$.

Let us formulate some definitions and properties of hierarchical data used in further proofs. From now on, names are considered as composite names from V^+ unless explicitly stated that they belong to V .

A *prefix* of a word $u \in V^+$ is any word x such that $u = x.y$ for some $y \in V^*$. If $u \neq x$, we call x a *strict prefix*. We write $x \leq u$ ($x < u$), if x is a prefix (strict prefix) of u . Words x and u are *comparable* ($x < > u$), if $x \leq u$ or $u \leq x$; otherwise they are *incomparable* ($x \div u$). Sets of names X and Y are *incomparable* ($X \div Y$), if $x \div y$ for all $x \in X$ and $y \in Y$.

We call a composite name a *full name* of d , if it coincides with some path from a root in the tree determined by d . If this path reaches a leaf, then the name is called *terminal*. We define the set of full names by $fn(d) = \{u \mid u : d \downarrow\}$; the set of terminal names by $tn(d) = \{u \mid u : d \downarrow \in A\}$.

Hierarchical data d_1 and d_2 are *disjoint*, if $x \div y$ for any $x \in tn(d_1)$ and $y \in tn(d_2)$. The union of disjoint data we denote by “+”.

Parametric operation of *deletion* of data components, the names of which are comparable with given names x_1, \dots, x_n , is defined via FNF as follows:

$$\|_{-x_1, \dots, x_n} (d) = [u \mapsto a \in d \mid u \text{ is terminal and } x_1 \div u, \dots, x_n \div u].$$

For basic data $a \in A$, $\|_{-x_1, \dots, x_n} (a)$ is undefined.

In the sequel instead of $\|_{-x_1, \dots, x_n} (d)$ we write $d \|_{-x_1, \dots, x_n}$.

Example 3. Let d be a hierarchical data from example 2. Then:

$$d \|_{-x, u} = [y.x \mapsto 3, y.y.x \mapsto 0, y.y.y \mapsto 0, y.y.z \mapsto 1, z \mapsto 2];$$

$$d \|_{-x.z, y.y, z.y, u.x, u} = [x.y \mapsto 1, y.x \mapsto 3, u.x.x \mapsto 0, u.z \mapsto 3].$$

When using the symbol “+” we drop brackets “[” and “]”, e.g. instead of $d \|_{-u} + [u \mapsto u : d \|_{-v}]$ we write $d \|_{-u} + u \mapsto u : d \|_{-v}$.

Proposition 1.

- 1) $d \|_{-z, u, x_1, \dots, x_n} = d \|_{-z, x_1, \dots, x_n}$, if $z \leq u$;
- 2) $d = d \|_{-x} + x \mapsto x : d$, if $x \in V$;
- 3) $(d \|_{-x}) \|_{-x, y} = (d \|_{-x, y}) \|_{-x} = d \|_{-x}$;
- 4) $(d_1 + d_2) \|_{-u} = d_1 \|_{-u} + d_2 \|_{-u}$;
- 5) $d \|_{-u, v} = d \|_{-u} + u \mapsto u : d \|_{-v}$;
- 6) $d \supseteq d \|_{-u} + u \mapsto u : d$.

For a composite u the property $d = d \|_{-u} + u \mapsto u : d$ may fail.

Example 4. Let $d = [u \mapsto 0, z.x \mapsto 0, z.y \mapsto 1]$. Then $d \|_{-u, v} = [z.x \mapsto 0, z.y \mapsto 1]$, and $d \|_{-u, v} + u.v \mapsto u.v : d = d \|_{-u, v} \neq d$, because $u : d$ is a basic value and $u.v : d$ is undefined.

Proposition 2. Let $u \div \{x_1, \dots, x_n\}$, then $d \|_{-u, x_1, \dots, x_n} = (d \|_{-u}) \|_{-x_1, \dots, x_n}$.

In particular, if $z \div u$, then $d \|_{-u, z} = (d \|_{-u}) \|_{-z} = (d \|_{-z}) \|_{-u}$.

In the general case we have that $d \|_{-u_1, \dots, u_m, x_1, \dots, x_n} = (d \|_{-u_1, \dots, u_m}) \|_{-x_1, \dots, x_n}$ if

$$\{u_1, \dots, u_m\} \div \{x_1, \dots, x_n\}.$$

Proposition 3. 1) $x : (d \|_{-z_1, \dots, z_n} + x \mapsto h) = h$; in particular, $x : (x \mapsto h) = h$;

2) $x : (d \|_{-z_1, \dots, z_n}) = x : d$, if $x \div \{z_1, \dots, z_n\}$;

3) $x : (d \|_{-x, z_1, \dots, z_n}) \uparrow$.

Using the propositions 1–3, it is possible to represent $d \parallel_{-x_1, \dots, x_n}$, where $x_1, \dots, x_n \in V^+$, with an expression in some standard form, which uses only operations of deletion $\parallel_{-v_1, \dots, v_m}$ with simple names $v_1, \dots, v_m \in V$, union $+$, naming $y_1.y_2. \dots .y_k \mapsto$ and denaming $u_1:u_2:\dots:u_l$: (here $y_1, \dots, y_k, u_1, \dots, u_l \in V$). Details are omitted here.

Example 5. $d \parallel_{-z.x, z.y, u.x.y} = d \parallel_{-z, u + z \mapsto z: d \parallel_{-x, y + u \mapsto u: d \parallel_{-x + u.x \mapsto x: u: d \parallel_{-y}}$.

Operation of renomination $r_{x_1, \dots, x_n}^{v_1, \dots, v_n} : HD(V, A) \rightarrow HD(V, A)$ we define as follows:

$$r_{x_1, \dots, x_n}^{v_1, \dots, v_n}(d) = d \parallel_{-v_1, \dots, v_n} + v_1 \mapsto x_1 : d + \dots + v_n \mapsto x_n : d .$$

Here all names v_1, \dots, v_n should be pairwise incomparable. We see that the result of renomination can be presented uniquely in the standard form.

Example 6. $r_{u.y, y.x, v.x}^{v.x, v.y, u.x.y}(d) = d \parallel_{-v, u + v \mapsto v: d \parallel_{-x, y + u \mapsto u: d \parallel_{-x +$

$$+ u.x \mapsto x: u: d \parallel_{-y} + v.x \mapsto y: u: d + v.y \mapsto x: y: d + u.x.y \mapsto x: v: d .$$

Note, that renomination is monotone: if $d \subseteq h$, then $r_{x_1, \dots, x_n}^{v_1, \dots, v_n}(d) \subseteq r_{x_1, \dots, x_n}^{v_1, \dots, v_n}(h)$.

To present convolution of renominations we use the standard form.

Example 7. $r_u^z(r_x^{u.v}(d)) = r_u^z(d \parallel_{-u.v} + u.v \mapsto x : d) =$

$$= r_u^z(d \parallel_{-u} + u \mapsto u : d \parallel_{-v} + u.v \mapsto x : d) =$$

$$d \parallel_{-u.z} + u.v \mapsto x : d + z \mapsto u : d \parallel_{-v} + z.v \mapsto x : d .$$

Thus, situation for hierarchical data is more difficult than for flat data for which convolution of renominations can be presented as one new renomination [3].

Example 8. $r_{z.y}^{x.v}(r_{x, x.y}^{u.v, z}(d)) =$

$$= r_{z.y}^{x.v}(d \parallel_{-u.z} + u \mapsto u : d \parallel_{-v} + u.v \mapsto x : d + z \mapsto y : x : d) =$$

$$= d \parallel_{-u.z, x} + u \mapsto u : d \parallel_{-v} + u.v \mapsto x : d + z \mapsto y : x : d + x \mapsto x : d \parallel_{-v} +$$

$$x.v \mapsto y : y : x : d .$$

3 Compositions of Predicates over Hierarchical Data

From semantic point of view the notion of predicate is one of the basic concepts of logic.

By a *predicate* P on D we understand a single-valued partial function of the type $D \rightarrow Bool$. The truth and falsity domains of P are respectively $T(P) = \{d \in D \mid P(d) \downarrow = T\}$ and $F(P) = \{d \in D \mid P(d) \downarrow = F\}$. A predicate P is *irrefutable*, or partially true, if $F(P) = \emptyset$.

Compositions determine universal methods of predicate construction; they form the kernel of logic of corresponding type.

At the *propositional* level data are treated as abstract, therefore predicates are interpreted as functions from A to $Bool$, where A is an abstract class. Basic propositional compositions are disjunction \vee and negation \neg ($P, Q \in A \rightarrow Bool, d \in A$):

$$(P \vee Q)(d) = \begin{cases} T, & \text{if } P(d) \downarrow = T \text{ or } Q(d) \downarrow = T, \\ F, & \text{if } P(d) \downarrow = F \text{ and } Q(d) \downarrow = F, \\ \text{undefined} & \text{in other cases.} \end{cases}$$

$$(\neg P)(d) = \begin{cases} T, & \text{if } P(d) \downarrow = F, \\ F, & \text{if } P(d) \downarrow = T, \\ \text{undefined,} & \text{if } P(d) \uparrow. \end{cases}$$

At the *nominative* level data are constructed from a set of subject names and a class of subject values. In this work logics of partial predicates over hierarchical nominative data at renominative and quantifier level are investigated.

A function of the form $P : HD(V, A) \rightarrow Bool$ is called a *hierary predicate* on $HD(V, A)$. We denote the class of hierary predicates on $HD(V, A)$ by PrH^{V-A} .

The name $x \in V$ is *strictly unessential* for a hierary predicate P on $HD(V, A)$, if for arbitrary $d, \alpha \in HD(V, A)$ we have $P(d \downarrow_{-x} + x \mapsto \alpha) = P(d \downarrow_{-x})$. The notion of unessential name is an analogue of fresh name in classical and nominal logics [15].

A predicate $P : HD(V, A) \rightarrow Bool$ is called *equitone*, if for arbitrary $d, d' \in HD(V, A)$ conditions $d \subseteq d'$ and $P(d) \downarrow$ imply $P(d') \downarrow = P(d)$.

At the renominative level to propositional compositions we add renomination composition $R_{x_1, \dots, x_n}^{v_1, \dots, v_n}$ defined by the formula

$$R_{x_1, \dots, x_n}^{v_1, \dots, v_n}(Q)(d) = Q(r_{x_1, \dots, x_n}^{v_1, \dots, v_n}(d)) = Q(d \downarrow_{-v_1, \dots, v_n} + v_1 \mapsto x_1 : d + \dots + v_n \mapsto x_n : d).$$

Using vector notation, we can formulate the following properties of renomination:

$$R \vee) R_{\bar{x}}^{\bar{v}}(P \vee Q) = R_{\bar{x}}^{\bar{v}}(P) \vee R_{\bar{x}}^{\bar{v}}(Q); R \neg) R_{\bar{x}}^{\bar{v}}(\neg P) = \neg R_{\bar{x}}^{\bar{v}}(P).$$

The properties of $R \rightarrow$, $R \&$, $R \leftrightarrow$ can be written down analogously.

$$RR) R_{\bar{x}}^{\bar{v}}(R_{\bar{y}}^{\bar{u}}(P)(d) = P(r_{\bar{y}}^{\bar{u}}(r_{\bar{x}}^{\bar{v}}(d)))) \text{ for each } d \in HD(V, A).$$

$$RSN) R_{z, \bar{x}}^{y, \bar{v}}(P) = R_{\bar{x}}^{\bar{v}}(P), \text{ if } y \in V \text{ is strictly unessential for } P.$$

$$RT) R_{z, \bar{x}}^{z, \bar{v}}(P) = R_{\bar{x}}^{\bar{v}}(P) \text{ under condition } z \in V.$$

In the case of equitone predicates for composite names we have:

$$RTE) R_{u, \bar{x}}^{u, \bar{v}}(P) \cong R_{\bar{x}}^{\bar{v}}(P), \text{ where } \cong \text{ is weak equality.}$$

At the *quantifier* level basic compositions are $\vee, \neg, R_{\bar{x}}^{\bar{v}}, \exists x$.

Contrary to traditional case quantified names can be composite; quantification is possible both over all hierarchical or only over basic data. In this work we consider quantification over hierarchical data. Composition of existential quantification is defined in the following way:

$$\exists x P(d) = \begin{cases} T, & \text{if there exists } \beta \in ND(V, A) : P(d \downarrow_{-x} + x \mapsto \beta) \downarrow = T, \\ F, & \text{if } P(d \downarrow_{-x} + x \mapsto \alpha) \downarrow = F \text{ for all } \alpha \in ND(V, A), \\ \text{undefined} & \text{in all other cases.} \end{cases}$$

Composition of universal quantification is defined by formula $\forall x P = \neg \exists x \neg P$.

Theorem 1. The class of equitone predicates over hierarchical data is closed under compositions $\vee, \neg, R_{\bar{x}}^{\bar{v}}, \exists x, \forall x$.

Main properties of compositions $\exists x$ and $\forall x$ are the following.

1. If x and y are incomparable then $\exists x\exists yP = \exists y\exists xP$ and $\forall x\forall yP = \forall y\forall xP$.
 2. Absorption of external quantifier by internal with the same name:
 $\exists x\exists xP = \exists xP$; $\forall x\exists xP = \exists xP$; $\exists x\forall xP = \forall xP$; $\forall x\forall xP = \forall xP$.
 3. Absorption of external quantifier by internal with more general name:
 $\exists x.y\exists xP = \exists xP$; $\forall x.y\exists xP = \exists xP$; $\exists x.y\forall xP = \forall xP$; $\forall x.y\forall xP = \forall xP$.
- At the same time $\exists x\exists x.yP$, $\forall x\exists x.yP$, $\exists xP$, $\exists x.yP$ are all different;
 $\exists x\forall x.yP$, $\forall x\forall x.yP$, $\forall xP$, $\forall x.yP$ are all different.
4. Absorption of the quantified name by more general upper name of renomination:
 $\exists x.y(R_{z,\bar{v}}^{x,\bar{u}}P) = R_{z,\bar{v}}^{x,\bar{u}}(P)$, if $x.y \div \{z, \bar{u}, \bar{v}\}$.
 5. Absorption of the upper name of renomination by more general quantifier:
 $R_{z,\bar{v}}^{\bar{y},\bar{u}}(\exists xP) = R_{\bar{v}}^{\bar{u}}(\exists xP)$, if x is a prefix of names from \bar{y} and $x \div \{\bar{u}\}$.

At the same time $\exists x.y(R_z^xP) \neq R_z^x(\exists x.yP)$ and $\exists x.y(R_{x.y.v}^xP) \neq R_{x.y.v}^x(P)$;

$$R_u^{x.y}(\exists xP) \neq \exists x(R_u^{x.y}(P)), R_u^{x.y}(\exists xP) \neq R_u^{x.y}(P), \exists x(R_{x.u}^zP) \neq R_{x.u}^z(\exists xP).$$

6. $\exists z(R_{\bar{v}}^{\bar{u}}P) = R_{\bar{v}}^{\bar{u}}(\exists zP)$, if $z \div \{\bar{u}, \bar{v}\}$.

Properties 4–6 can be rephrased for universal quantification.

Let us note that some properties valid in classical logic fail for the class of equitone predicates over hierarchical data.

Example 9. Let predicate τ_x be defined by the following formula:

$$\tau_x(d) = \begin{cases} T, & \text{if } d(x) \downarrow \in A, \\ F, & \text{if } d(x) \downarrow \notin A, \\ \text{undefined} & \text{in all other cases.} \end{cases}$$

It is clear that τ_x is equitone. By definitions of compositions $\exists x$ and $\forall xP$ we have that $\exists x.v \tau_x(d) = \forall x.v \tau_x(d) = F$ for each $d \in HD(V, A)$ such that $x \rightarrow a \in d$, where $a \in A$. At the same time $\tau_x(d) = T$ for such d . So, $(\tau_x \rightarrow \exists x.v \tau_x)(d) = F$.

4 Semantic Models and Languages of Logics over Hierarchical Data

Semantic models of *composition-nominative logics over hierarchical nominative data* (CNLH) are predicate algebras with class $PrH^{V,A}$ of hierary predicates as carriers and class \mathcal{C} of compositions as operations of algebras. The class \mathcal{C} is determined by a level intensional; for a quantifier level \mathcal{C} consists of compositions $\vee, \neg, R_{\bar{x}}^{\bar{y}}$, and $\exists x$; for renominative level these are \vee, \neg , and $R_{\bar{x}}^{\bar{y}}$. Thus, algebras of the form $AHD(V, A) = \langle PrH^{V,A}, \vee, \neg, R_{\bar{x}}^{\bar{y}}, \exists x \rangle$ are semantic base of constructed logics. With a fixed sets V and \mathcal{C} such algebras are determined by the set A .

Alphabet of a language of quantifier level includes symbols of basic compositions, a set Ps of *predicate symbols*, and a set of *basic subject names (variables)* V .

The set Fr of formulas for a quantifier level is defined inductively:

- 1) every predicate symbol from Ps is an (atomic) formula;

- 2) if Φ and Ψ are formulas, then $\Phi \vee \Psi$ and $\neg \Phi$ are formulas;
- 3) if Φ is a formula, then $R_{\bar{x}}^V \Phi$ is a formula;
- 4) if Φ is a formula, then $\exists x \Phi$ is a formula.

For CNLH of renominative level we drop item 4 in this definition.

Let $nm(\Phi)$ be the set of all names, which appear in the symbols of renomination and quantification in Φ .

To distinguish symbols of compositions from their interpretations we use for the latter bold font (only in the following definitions). Let $\mathbf{I} : Ps \rightarrow PrH^{V-A}$ be a total single-valued *interpretation mapping*, then a pair $(AHD(V, A), \mathbf{I})$ is called a *model of CNLH language*. To simplify notation we will denote models as (A, \mathbf{I}) Interpretation $\mathbf{J} : Fr \rightarrow PrH^{V-A}$ we define as follows:

- 1) $\mathbf{J}(p) = \mathbf{I}(p)$ for each $p \in Ps$;
- 2) $\mathbf{J}(\Phi \vee \Psi) = \mathbf{J}(\Phi) \vee \mathbf{J}(\Psi)$, $\mathbf{J}(\neg \Phi) = \neg(\mathbf{J}(\Phi))$;
- 3) $\mathbf{J}(R_{\bar{x}}^V \Phi) = R_{\bar{x}}^V(\mathbf{J}(\Phi))$;
- 4) $\mathbf{J}(\exists x \Phi) = \exists x(\mathbf{J}(\Phi))$.

For renominative level we drop item 4.

Predicate $\mathbf{J}(\Phi)$, which is the value of a formula Φ interpreted on $A = (A, \mathbf{I})$, we denote by Φ_A . A formula Φ is partially true on $A = (A, \mathbf{I})$ (denoted by $A \models \Phi$), if Φ_A is partially true (irrefutable) predicate. Φ is everywhere (partially) true, or irrefutable (denoted by $\models \Phi$), if Φ is partially true on every model of a language.

A formula Ψ is a *logical consequence* of a formula Φ ($\Phi \models \Psi$), if formula $\Phi \rightarrow \Psi$ is irrefutable. Ψ is a *weak logical consequence* of Φ ($\Phi \Vdash \Psi$), if for each $A = (A, \mathbf{I})$ the condition $A \models \Phi$ implies $A \models \Psi$.

Formulas Φ and Ψ are *logically equivalent* ($\Phi \sim \Psi$), if $\Phi \models \Psi$ and $\Psi \models \Phi$. Formulas Φ and Ψ are *logically strictly equivalent* ($\Phi \sim_{TF} \Psi$), if $\mathbf{T}(\Phi_A) = \mathbf{T}(\Psi_A)$ and $\mathbf{F}(\Psi_A) = \mathbf{F}(\Phi_A)$ for each AS A . The relation of logical consequence can be extended to arbitrary sets $\Gamma, \Delta \subseteq Fr$. Δ is a logical consequence of Γ in the model A ($\Gamma_A \models \Delta$) if for all $d \in HD(V, A)$ the condition $\Phi_A(d) \downarrow = T$ for all $\Phi \in \Gamma$ implies that it is impossible that $\Psi_A(d) \downarrow = F$ for all $\Psi \in \Delta$. Δ is a logical consequence of Γ ($\Gamma \models \Delta$), if $\Gamma_A \models \Delta$ for all model $A = (A, \mathbf{I})$. Relation \models is reflective but not transitive.

For CNLH the following statements hold.

Theorem 2 (semantic equivalence). Suppose that Φ' is obtained from Φ by substitution of some occurrences of Φ_1, \dots, Φ_n with Ψ_1, \dots, Ψ_n respectively. If $\Phi_1 \sim \Psi_1, \dots, \Phi_n \sim \Psi_n$, then $\Phi \sim \Phi'$.

Theorem 3 (semantic equivalence, strong form). Suppose that Φ' is obtained from Φ by substitution of some occurrences of Φ_1, \dots, Φ_n with Ψ_1, \dots, Ψ_n respectively. If $\Phi_1 \sim_{TF} \Psi_1, \dots, \Phi_n \sim_{TF} \Psi_n$, then $\Phi \sim_{TF} \Phi'$.

Theorem 4 (substitution of equivalents). Suppose that $\Phi \sim \Psi$. Then $\Phi, \Gamma \models \Delta \Leftrightarrow \Psi, \Gamma \models \Delta$ and $\Gamma \models \Delta, \Phi \Leftrightarrow \Gamma \models \Delta, \Psi$.

A name $x \in V$ is *strictly unessential* for Φ ($x \in sun(\Phi)$), if x is strictly unessential for a predicate Φ_A for every $A = (A, \mathbf{I})$.

Proposition 4. Let $y \in sun(\Phi)$. Then $\exists x \Phi \sim_{TF} \exists y R_y^x \Phi$.

For each $p \in Ps$ the set of strictly unessential subject names is fixed by a total function $v: Ps \rightarrow 2^V$. For CNLH we postulate infinity of the set $V_T = \bigcap_{p \in Ps} v(p)$ of

totally strictly unessential names.

The following properties of formulas are representations of corresponding semantic properties of predicate algebras.

$$\text{RsN)} R_{z,\bar{x}}^{y,\bar{v}}(\Phi) \sim_{TF} R_{\bar{x}}^{\bar{v}}(\Phi), \text{ if } y \in \text{sun}(\Phi).$$

$$\text{RT)} R_{z,\bar{x}}^{z,\bar{v}}(\Phi) \sim_{TF} R_{\bar{x}}^{\bar{v}}(\Phi), \text{ if } z \in V; \text{ in particular, } R_z^z(\Phi) \sim_{TF} \Phi.$$

$$\text{R}\vee) R_{\bar{x}}^{\bar{v}}(\Phi \vee \Psi) \sim_{TF} R_{\bar{x}}^{\bar{v}}(\Phi) \vee R_{\bar{x}}^{\bar{v}}(\Psi).$$

$$\text{R}\neg) R_{\bar{x}}^{\bar{v}}(\neg\Phi) \sim_{TF} \neg R_{\bar{x}}^{\bar{v}}(\Phi).$$

Generalizing R \vee and R \neg , we get RR \vee and RR \neg .

$$\text{RR}\vee) R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(\Phi \vee \Psi) \dots) \sim_{TF} R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(\Phi) \dots) \vee R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(\Psi) \dots).$$

$$\text{RR}\neg) R_{\bar{x}}^{\bar{u}}(R_{\bar{y}}^{\bar{v}}(\dots R_{\bar{z}}^{\bar{w}}(\neg\Phi) \dots)) \sim_{TF} \neg R_{\bar{x}}^{\bar{u}}(R_{\bar{y}}^{\bar{v}}(\dots R_{\bar{z}}^{\bar{w}}(\Phi) \dots)).$$

Similarly, we can write down the properties R $\&$, R \rightarrow , R \leftrightarrow , RR $\&$, RR \rightarrow , RR \leftrightarrow .

$$\text{RR}_C) R_{\bar{x}}^{\bar{v}}(R_{\bar{y}}^{\bar{u}}(\Phi_A)(d) = \Phi_A(r_{\bar{y}}^{\bar{u}}(r_{\bar{x}}^{\bar{v}}(d)))) \text{ for each } A = (A, \mathbf{I}), d \in HD(V, A).$$

$$\text{ANQ)} \exists x.y(R_{z,\bar{v}}^{x,\bar{u}}\Phi) \sim_{TF} R_{z,\bar{v}}^{x,\bar{u}}(\Phi) \text{ and } \forall x.y(R_{z,\bar{v}}^{x,\bar{u}}\Phi) \sim_{TF} R_{z,\bar{v}}^{x,\bar{u}}(\Phi), \text{ if } x.y \div \{z, \bar{u}, \bar{v}\}.$$

ANR) $R_{\bar{z},\bar{v}}^{\bar{y},\bar{u}}(\exists x\Phi) \sim_{TF} R_{\bar{v}}^{\bar{u}}(\exists x\Phi)$ and $R_{\bar{z},\bar{v}}^{\bar{y},\bar{u}}(\forall x\Phi) \sim_{TF} R_{\bar{v}}^{\bar{u}}(\forall x\Phi)$, if x is a prefix of all names in \bar{y} and $x \div \{\bar{u}\}$.

$$\text{RE}\exists) R_{\bar{x}}^{\bar{v}}(\exists y\Phi) \sim_{TF} \exists y(R_{\bar{x}}^{\bar{v}}\Phi), \text{ if } y \div \{\bar{u}, \bar{v}\}.$$

$$\text{RE}\exists\exists) R_{\bar{x}}^{\bar{v}}(\exists y\Phi) \sim_{TF} \exists z R_{\bar{x}}^{\bar{v}}(R_{\bar{z}}^{\bar{y}}(\Phi)) \text{ if } z \in V_T \text{ and } z \notin nm(R_{\bar{x}}^{\bar{v}}(\exists y\Phi)).$$

Similarly, we can formulate R \forall and R $\forall\forall$. Properties R \exists , R $\exists\exists$, R \forall , R $\forall\forall$ can be generalized to RR \exists , RR $\exists\exists$, RR \forall , RR $\forall\forall$; R \vee and R \neg to RR \vee and RR \neg .

For equitone predicates RT can be changed to RTE:

$$\text{RTE)} R_{u,\bar{x}}^{u,\bar{v}}(\Phi) \sim R_{\bar{x}}^{\bar{v}}(\Phi); \text{ in particular } R_u^u(\Phi) \sim \Phi.$$

For logics of equitone predicates we introduce the notion of *primitive formula*. A formula $R_{\bar{x}}^{\bar{u}}(R_{\bar{y}}^{\bar{v}}(\dots R_{\bar{z}}^{\bar{w}}(p) \dots))$ is primitive, if $p \in Ps$ and in renominations identical pairs of names are removed.

With every primitive $R_{\bar{x}}^{\bar{u}}(R_{\bar{y}}^{\bar{v}}(\dots R_{\bar{z}}^{\bar{w}}(p) \dots))$ we connect an expression of the form $p(\alpha)$, where α represents a convolution of renominations $r_{\bar{x}}^{\bar{u}}(r_{\bar{y}}^{\bar{v}}(\dots r_{\bar{z}}^{\bar{w}}(\delta) \dots))$ given in the standard form, $\delta \notin V \cup Ps$ is a special symbol, which denotes arbitrary data. To take into account strictly unessential subject names, we delete all components that have $z \in v(p)$ as a prefix. An expression $p(\alpha)$ is called a *renominant* of the above primitive formula. The set of longest incomparable names occurred in a renominant is called its *naming scheme*.

Example 10. To construct the renominant of a primitive formula $R_x^{u,v}(R_u^z(q))$ we specify corresponding standard form of renomination convolution (see Example 7)

obtaining renominant $q(d \parallel_{-u,z} + u.v \mapsto x : d + z \mapsto u : d \parallel_{-v} + z.v \mapsto x : d)$. Its naming scheme is $\{u.v, x, z.v\}$.

Now we point out basic properties of quantification compositions for CNLH.

Q1. $\exists x \exists y \Phi \sim_{TF} \exists y \exists x \Phi$ and $\forall x \forall y \Phi \sim_{TF} \forall y \forall x \Phi$, if x and y are incomparable.

Q2. $\neg \forall x \Phi \sim_{TF} \exists x \neg \Phi$ and $\neg \exists x \Phi \sim_{TF} \forall x \neg \Phi$.

Q3. $\exists x \Phi \sim_{TF} \forall x \exists x \Phi$, $\exists x \Phi \sim_{TF} \exists x \exists x \Phi$; $\forall x \Phi \sim_{TF} \forall x \forall x \Phi$, $\forall x \Phi \sim_{TF} \exists x \forall x \Phi$.

Q4. $\exists x \Phi \sim_{TF} \forall x.y \exists x \Phi$, $\exists x \Phi \sim_{TF} \exists x.y \exists x \Phi$; $\forall x \Phi \sim_{TF} \forall x.y \forall x \Phi$, $\forall x \Phi \sim_{TF} \exists x.y \forall x \Phi$.

Q5. $\exists x \Phi \vee \exists x \Psi \sim_{TF} \exists x(\Phi \vee \Psi)$ and $\forall x \Phi \& \forall x \Psi \sim_{TF} \forall x(\Phi \& \Psi)$.

Q6. $\exists x(\Phi \& \Psi) \models \exists x \Phi \& \exists x \Psi$ and $\forall x \Phi \vee \forall x \Psi \models \forall x(\Phi \vee \Psi)$.

Q7. $\exists y \forall x \Phi \models \forall x \exists y \Phi$; and not always $\forall x \exists y \Phi \models \exists y \forall x \Phi$.

Q8. $\Phi \models \exists x \Phi$ and $\Phi \models \forall x \Phi$.

Q9. $\models \forall x(\forall x \Phi \rightarrow \Phi)$ and $\models \exists x(\forall x \Phi \rightarrow \Phi)$; $\models \forall x(\Phi \rightarrow \exists x \Phi)$ and $\models \exists x(\Phi \rightarrow \exists x \Phi)$.

Properties Q2, Q3, Q5–Q9 are analogous to the corresponding properties of logics of quasiary predicates.

At the propositional level the properties of \models for sets of formulas are identical to corresponding properties of logic of quasiary predicates [3].

Now we formulate basic properties of renomination compositions.

$\text{RTE}_{|-}$ $R_{u,\bar{x}}^{u,\bar{v}}(\Phi), \Gamma_A \models \Delta \Leftrightarrow R_{\bar{x}}^{\bar{v}}(\Phi), \Gamma_A \models \Delta$.

$\text{RTE}_{|-}$ $\Gamma_A \models \Delta, R_{u,\bar{x}}^{u,\bar{v}}(\Phi) \Leftrightarrow \Gamma_A \models \Delta, R_{\bar{x}}^{\bar{v}}(\Phi)$.

$\text{RsN}_{|-}$ $R_{z,\bar{x}}^{y,\bar{v}}(\Phi), \Gamma_A \models \Delta \Leftrightarrow R_{\bar{x}}^{\bar{v}}(\Phi), \Gamma_A \models \Delta$, where $y \in V$ is strictly unessential for Φ .

$\text{RsN}_{|-}$ $\Gamma_A \models \Delta, R_{z,\bar{x}}^{y,\bar{v}}(\Phi) \Leftrightarrow \Gamma_A \models \Delta, R_{\bar{x}}^{\bar{v}}(\Phi)$, where $y \in V$ is strictly unessential for Φ .

$\text{RR}_{\vee|-}$ $R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{v}}(\Phi \vee \Psi) \dots), \Gamma_A \models \Delta \Leftrightarrow R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{v}}(\Phi) \dots) \vee R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{v}}(\Psi) \dots), \Gamma_A \models \Delta$.

$\text{RR}_{\vee|-}$ $\Gamma, R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{v}}(\Phi \vee \Psi) \dots)_A \models \Delta \Leftrightarrow \Gamma, R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{v}}(\Phi) \dots) \vee R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{v}}(\Psi) \dots)_A \models \Delta$.

$\text{RR}_{\neg|-}$ $R_{\bar{x}}^{\bar{u}}(R_{\bar{y}}^{\bar{v}}(\dots R_{\bar{z}}^{\bar{w}}(\neg \Phi) \dots)), \Gamma_A \models \Delta \Leftrightarrow \neg R_{\bar{x}}^{\bar{u}}(R_{\bar{y}}^{\bar{v}}(\dots R_{\bar{z}}^{\bar{w}}(\Phi) \dots)), \Gamma_A \models \Delta$.

$\text{RR}_{\neg|-}$ $\Gamma, R_{\bar{x}}^{\bar{u}}(R_{\bar{y}}^{\bar{v}}(\dots R_{\bar{z}}^{\bar{w}}(\neg \Phi) \dots))_A \models \Delta \Leftrightarrow \Gamma, \neg R_{\bar{x}}^{\bar{u}}(R_{\bar{y}}^{\bar{v}}(\dots R_{\bar{z}}^{\bar{w}}(\Phi) \dots))_A \models \Delta$.

Properties $\text{RR}_{\rightarrow|-}$, $\text{RR}_{\rightarrow|-}$, $\text{RR}_{\&|-}$, $\text{RR}_{\&|-}$ are analogous.

$\text{RE}_{|-}$ $R_{\bar{x}}^{\bar{v}}(\exists y \Phi), \Gamma_A \models \Delta \Leftrightarrow \exists y R_{\bar{x}}^{\bar{v}}(\Phi), \Gamma_A \models \Delta$ if $y \div \{\bar{u}, \bar{v}\}$.

$\text{RE}_{|-}$ $\Gamma_A \models \Delta, R_{\bar{x}}^{\bar{v}}(\exists y \Phi) \Leftrightarrow \Gamma_A \models \Delta, \exists y R_{\bar{x}}^{\bar{v}}(\Phi)$ if $y \div \{\bar{u}, \bar{v}\}$.

$\text{RE}_{\exists|-}$ $R_{\bar{x}}^{\bar{v}}(\exists y \Phi), \Gamma_A \models \Delta \Leftrightarrow \exists z R_{\bar{x}}^{\bar{v}}(R_{\bar{z}}^y(\Phi)), \Gamma_A \models \Delta$.

$\text{RE}_{\exists|-}$ $\Gamma_A \models \Delta, R_{\bar{x}}^{\bar{v}}(\exists y \Phi) \Leftrightarrow \Gamma_A \models \Delta, \exists z R_{\bar{x}}^{\bar{v}}(R_{\bar{z}}^y(\Phi))$.

For $\text{RE}_{\exists|-}$ and $\text{RE}_{\exists|-}$ z is totally strictly unessential and $z \notin nm(R_{\bar{x}}^{\bar{v}}(\exists y \Phi))$.

Properties $\text{RV}_{|-}$, $\text{RV}_{|-}$, $\text{RV}_{\vee|-}$, $\text{RV}_{\vee|-}$ are analogous. Properties of type RE , RE_{\exists} , RV , RV_{\vee} can be generalized to properties of type RR_{\exists} , RR_{\exists} , RR_{\vee} , RR_{\vee} .

5 The Sequent Calculus of Logics of Predicates over Hierarchical Data

For logics of equitone hierary predicates we will build a calculus of sequent type. We will consider here only logics of renominative level. Sequents are interpreted as sets of *labeled formulas* marked by one of two symbols \vdash or \dashv . Such sequents Σ are also denoted by $\vdash \Gamma \dashv \Delta$, where all formulas of Γ are labeled by the symbol \vdash , of Δ – by the symbol \dashv .

Sequent Σ is *closed*, if there exists Φ such that $\vdash \Phi \in \Sigma$ and $\dashv \Phi \in \Sigma$ or if there exist primitive φ and ψ with identical renominants such that $\vdash \varphi \in \Sigma$ and $\dashv \psi \in \Sigma$. Consequently, if $\vdash \Gamma \dashv \Delta$ is closed then $\Gamma \models \Delta$.

Derivation in the sequent calculus has the form of tree, the vertices of which are sequents. Such trees [3] are called sequent trees. A sequent tree is *closed*, if every its leaf is a closed sequent. A sequent Σ is *derivable*, if there is a closed sequent tree with root Σ . Sequent calculus is constructed in such a way that sequent $\vdash \Gamma \dashv \Delta$ has a derivation if and only if $\Gamma \models \Delta$.

Semantic properties of relation \models have their syntactic analogues – sequent forms (rules). For renominative logics of equitone hierary predicates these forms are the following.

$$\begin{array}{l}
\vdash \vee \frac{\vdash A, \Sigma \quad \vdash B, \Sigma}{\vdash A \vee B, \Sigma} \\
\vdash \neg \frac{\dashv A, \Sigma}{\vdash \neg A, \Sigma} \\
\vdash \text{RTE} \frac{\vdash R_{\bar{x}}^{\bar{v}}(A), \Sigma}{\vdash R_{u, \bar{x}}^{u, \bar{v}}(A), \Sigma} \\
\vdash \text{RR}\vee \frac{\vdash R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(A)\dots) \vee R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(B)\dots), \Sigma}{\vdash R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(A \vee B)\dots), \Sigma} \\
\vdash \text{RR}\neg \frac{\vdash \neg R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(A)\dots), \Sigma}{\vdash R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(\neg A)\dots), \Sigma} \\
\vdash \vee \frac{\dashv A, \dashv B, \Sigma}{\dashv A \vee B, \Sigma} \\
\vdash \neg \frac{\vdash A, \Sigma}{\dashv \neg A, \Sigma} \\
\vdash \text{RTE} \frac{\dashv R_{\bar{x}}^{\bar{v}}(A), \Sigma}{\dashv R_{u, \bar{x}}^{u, \bar{v}}(A), \Sigma} \\
\vdash \text{RR}\vee \frac{\dashv R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(A)\dots) \vee R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(B)\dots), \Sigma}{\dashv R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(A \vee B)\dots), \Sigma} \\
\vdash \text{RR}\neg \frac{\dashv \neg R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(A)\dots), \Sigma}{\dashv R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(\neg A)\dots), \Sigma}
\end{array}$$

Sequent calculus with basic sequent forms shown above we will call *RID-calculus*. For *RID-calculus* theorems of soundness and completeness hold.

Theorem 5 (soundness). Let sequent $\vdash \Gamma \dashv \Delta$ be derivable. Then $\Gamma \models \Delta$.

The proof can be conducted by induction over shape of a sequent tree for $\vdash \Gamma \dashv \Delta$.

For proving completeness we will use *Hintikka's method of model sets*. The set H of labeled formulas with $W = nm(H)$ is a model set, if:

HC Φ) For every non-primitive formula Φ it is impossible that $\vdash \Phi \in H$ and $\dashv \Phi \in H$.

HCR) For primitive formulas φ and ψ with identical renominants it is impossible that $\vdash \varphi, \dashv \psi \in H$ and it is impossible that $\vdash \psi, \dashv \varphi \in H$.

- H \vee) If $\perp \Phi \vee \Psi \in \mathbf{H}$, then $\perp \Phi \in \mathbf{H}$ or $\perp \Psi \in \mathbf{H}$; if $\neg \Phi \vee \Psi \in \mathbf{H}$, then $\neg \Phi \in \mathbf{H}$ and $\neg \Psi \in \mathbf{H}$.
H \neg) If $\perp \neg \Phi \in \mathbf{H}$, then $\neg \Phi \in \mathbf{H}$; if $\neg \neg \Phi \in \mathbf{H}$, then $\perp \Phi \in \mathbf{H}$.
HRT) If $\perp R_{u,\bar{x}}^{\bar{u},\bar{v}}(\Phi) \in \mathbf{H}$, then $\perp R_{\bar{x}}^{\bar{v}}(\Phi) \in \mathbf{H}$; if $\neg R_{u,\bar{x}}^{\bar{u},\bar{v}}(\Phi) \in \mathbf{H}$, then $\neg R_{\bar{x}}^{\bar{v}}(\Phi) \in \mathbf{H}$.
HR \vee) If $\perp R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(\Phi \vee \Psi)\dots) \in \mathbf{H}$, then $\perp R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(\Phi)\dots) \vee R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(\Psi)\dots) \in \mathbf{H}$;
if $\neg R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(\Phi \vee \Psi)\dots) \in \mathbf{H}$, then

$$\neg R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(\Phi)\dots) \vee R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(\Psi)\dots) \in \mathbf{H}$$
.
HR \neg) If $\perp R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(\neg \Phi)\dots) \in \mathbf{H}$, then $\perp \neg R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(\Phi)\dots) \in \mathbf{H}$;
if $\neg R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(\neg \Phi)\dots) \in \mathbf{H}$, then $\neg \neg R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(\Phi)\dots) \in \mathbf{H}$.

Procedure of construction of a tree for Σ is split into stages. Every application of sequent form is performed only for the finite set of accessible formulas. At the beginning of every stage we perform the step of access: to the list of accessible formulas one formula from each of lists of \perp -formulas and \neg -formulas is added. We start the construction with a pair of first formulas from the lists.

Suppose that k stages of procedure have already been performed. On the stage $k+1$ we check whether all terminal nodes are closed. If yes, the procedure is completed positively, and we have got a closed sequent tree. If no, for every unclosed leaf ξ we undertake a next step of access, whereupon we finish building of finite subtree with a vertex ξ as follows.

We activate all accessible non-primitive formula ξ . Then to every active formula we apply the proper sequent form. We remove all repetitions of formulas in a sequent.

During the construction of sequent tree the following cases are possible:

1. Procedure is completed positively; we have the finite closed tree.
2. Procedure is completed negatively, or is not completed; we have a finite

or infinite unclosed tree. Such tree has at least one path all vertices of which are unclosed sequents. Such path \wp is unclosed. Every formula of Σ will be in \wp and will become accessible.

Theorem 6. Let \wp be an unclosed path in sequent tree. Then there exists AS $A = (A, \mathbf{I})$ and $\delta \in HD(V, A)$: $\perp \Phi \in \mathbf{H} \Rightarrow \Phi_A(\delta) \downarrow = T$ and $\neg \Phi \in \mathbf{H} \Rightarrow \Phi_A(\delta) \downarrow = F$.

The set \mathbf{H} of labeled formulas of sequents of the path \wp is a model set.

Let W be a combination of naming schemes of the set of renominants of primitive formulas of sequents of the path \wp . Such W includes longest incomparable names, which are involved in renominations of formulas of sequents of the path \wp .

We duplicate elements of W obtaining $A = \{u \mid u \in W\}$; then put $\delta = [u \rightarrow \bar{u} \mid u \in W]$.

We specify the values of basic predicates on δ and on data of the form $r_{\bar{x}}^{\bar{u}}(\dots r_{\bar{z}}^{\bar{w}}(\delta)\dots)$ in the following way:

- if $\perp p \in \mathbf{H}$, then set $p_A(\delta) = T$; if $\neg p \in \mathbf{H}$, then set $p_A(\delta) = F$;
- if $\perp R_{\bar{x}}^{\bar{u}}(\dots R_{\bar{z}}^{\bar{w}}(p)\dots) \in \mathbf{H}$, then set $p_A(r_{\bar{x}}^{\bar{u}}(\dots r_{\bar{z}}^{\bar{w}}(\delta)\dots)) = T$;
- if $\neg R_{\bar{x}}^{\bar{u}}(R_{\bar{y}}^{\bar{v}}(\dots R_{\bar{z}}^{\bar{w}}(p)\dots)) \in \mathbf{H}$, then set $p_A(r_{\bar{x}}^{\bar{u}}(r_{\bar{y}}^{\bar{v}}(\dots r_{\bar{z}}^{\bar{w}}(\delta)\dots))) = F$.

In all other cases for $d \in HD(V, A)$ the value of $p_A(d)$ can be set arbitrarily, taking into account equitonicity and strict inessentiality of names.

Theorem holds for atomic and primitive formulas due to above definitions of basic

predicates. Then the proof is carried out by induction over the complexity of a formula in accordance with construction of a model set.

Theorem 7 (completeness). Let $\Gamma \models \Delta$. Then a sequent $\perp \Gamma \perp \Delta$ is derivable.

Suppose contrary: $\Gamma \models \Delta$ and a sequent $\perp \Gamma \perp \Delta$ is not derivable. Then sequent tree δ for $\Sigma = \perp \Gamma \perp \Delta$ is not closed. Consequently, in δ there is unclosed path \wp . The set H of all labeled formulas of sequents of this path is a model set. According to the theorem 5 there exists AS $A = (A, I)$ and $\delta \in HD(V, A)$ such that $\perp \Phi \in H \Rightarrow \Phi_A(\delta) \downarrow = T$ and $\perp \Phi \in H \Rightarrow \Phi_A(\delta) \downarrow = F$. Due to $\Sigma \subseteq H$ we have $\perp \Phi \in \Sigma \Rightarrow \Phi_A(\delta) \downarrow = T$ and $\perp \Phi \in \Sigma \Rightarrow \Phi_A(\delta) \downarrow = F$. But it contradicts $\Gamma \models \Delta$.

6 Conclusions

In the paper new logics oriented on hierarchical data are developed. Algebras of partial predicates over such data with special compositions as operations form a semantic base for constructed logics. These logics may also be treated as generalization of classical logic. First of all, this generalization concerns types of predicates: while classical logic is semantically based on total n -ary predicates, we have constructed logics based on partial quasiary and hierary predicates, defined on special types of hierarchical nominative data. Importance of such data is explained by their representational power, which permits to model data structures of specification and programming languages. Characteristic feature of such languages is usage of composite names to access data components. The constructed logics also use composite names. Semantic properties of such logics have been studied; corresponding sequent calculi have been defined, their soundness and completeness have been proved for logics of renominative level. Authors plan to present more developed logics at hierarchical nominative level in forthcoming papers.

References

1. Nikitchenko, N.S.: A Composition-nominative Approach to Program Semantics. Technical Report IT-TR 1998-020, Technical University of Denmark, 103 p. (1998)
2. Nikitchenko, M.S.: Composition-nominative aspects of address programming. Cybernetics and Systems Analysis, No. 6, pp. 24-35 (2009) (In Russian). English translation: Springer New York, Volume 45, Number 6 / November, 2009.
3. Nikitchenko, M.S., Shkilniak, S.S.: Mathematical logic and theory of algorithms. Publishing house of National Taras Shevchenko University of Kyiv, 528 p. (2008) (in Ukrainian).
4. Basarab, I.A., Gubsky, B.V., Nikitchenko, N.S., Red'ko, V.N.: Composition models of databases. In: Eder, J., Kalinichenko, L.A. (eds.) East-West Database Workshop.- (Workshops in Computing Series). Springer, London, pp. 221-231 (1995)
5. Nikitchenko, N.S.: Abstract Computability of Non-deterministic Programs over Various Data Structures. In: Bjørner, D., Broy, M., Zamulin A.V. (eds.) Perspectives of System Informatics. LNCS, vol. 2244, pp. 471-484. Springer, Berlin (2001)
6. Shkilniak, S.S.: Relations of logical consequence in composition-nominative logics. Problems of Programming. Kyiv, No. 1, pp. 15-38, 2010 (In Ukrainian)

7. Nikitchenko, M.S., Shkilnyak, S.S., Omelchuk, L.L.: Formalisms for Specification of Programs over Nominative Data. In: Electronic computers and informatics (ECI 2006). Thesis of conference reports, pp. 134-139. Kosice, Herl'any, Slovakia (2006)
8. Nielson, H.R., Nielson, F.: Semantics with Applications: A Formal Introduction. John Wiley & Sons Inc. 252 p. (1992)
9. Kleene, S. C.: Introduction to metamathematics, Van Nostrand, New York (1952)
10. Woodcock, J.C.P., Davies, J.: Using Z: Specification, Refinement and Proof. Prentice Hall, 523 p. (1996)
11. Abrial, J.R.: The B-Book: Assigning programs to meanings. Cambridge University Press, 779 p. (1996)
12. Lamport, L.: Specifying Systems: The TLA⁺ Language and Tools for Hardware and Software Engineers. Addison-Wesley (2002)
13. Lamport, L.: Substitution: Syntactic versus Semantic SRC Technical Note 1998-004 (March 1998)
14. Nikitchenko, M.S., Shkilnyak, S.S., Composition-nominative logics over hierarchical data. Problems of Programming. Kyiv, No. 2-3, pp. 48–57, 2010 (In Ukrainian)
15. Pitts, A.M.: Nominal logic, a first order theory of names and binding. Inf. Comput. 186(2), pp. 165-193 (2003)

On Existence of Global-in-Time Trajectories of Non-deterministic Markovian Systems

Ievgen Ivanov^{1,2}

¹Taras Shevchenko National University of Kyiv, Ukraine

²Paul Sabatier University, Toulouse, France
ivanov.eugen@gmail.com

Abstract. We consider the following question: given a continuous-time non-deterministic (not necessarily time-invariant) dynamical system, is it true that for each initial condition there exists a global-in-time trajectory. We study this question for a large class of systems, namely the class of complete non-deterministic Markovian systems. We show that for this class of systems, the question can be answered using analysis of existence of locally defined trajectories in a neighborhood of each time.

Keywords. dynamical systems, non-deterministic systems, Markovian systems, global-in-time trajectories

Key Terms. Mathematical Model, Specification Process, Verification Process

1 Introduction

In this paper we consider the following question: given a continuous-time non-deterministic (not necessarily time-invariant) dynamical system Σ , is it true that for any time moment t_0 and initial state x_0 there exists a global-in-time trajectory $t \mapsto s(t)$ such that $s(t_0) = x_0$.

Some related problems, e.g. global existence of solutions of initial value problems for various classes of differential equations [2, 3, 7] and inclusions [4–6], existence of non-Zeno global-in-time executions of hybrid automata [8–10] are well known. However, they have mostly been studied in the context of deterministic systems (differential equations with unique solutions, deterministic hybrid automata, etc.). Differential inclusions [5] are in principle non-deterministic systems, but for them a more common question is whether any (instead of some) solution for each initial condition exists into future [4, 6].

For deterministic systems the existence of a global trajectory for each initial condition implies that each partial trajectory (e.g. defined on a proper open interval of the real time scale) can be extended to a global trajectory. But this is not necessary for non-deterministic systems. For example, for each initial condition $x(t_0) = x_0$ the differential inclusion $\frac{dx}{dt} \in [0, x^2]$ has both a globally defined

constant trajectory $x(t) = x_0$ and a trajectory of the equation $\frac{dx}{dt} = x^2$ which escapes to infinity in finite time. Thus it is not true that any (locally defined) solution extends infinitely into future.

We will study our existence question for a large class of systems, namely the class of complete non-deterministic Markovian systems. We will show that for this class of systems, the question can be answered using analysis of existence of locally defined trajectories in a neighborhood of each time.

Note that in this paper we use the term Markovian in the context of purely non-deterministic (i.e. non-stochastic) systems. The formal definition will be given below. Also note that many well-known classes of continuous-time systems either belong to this class or can be represented by systems of this class. We will give examples later in the paper.

2 Non-deterministic Complete Markovian Systems

The notions of a Markov process or system [12] are usually defined and studied in the context of probability theory. However, they also make sense in a purely non-deterministic setting, where no quantitative information is attached to events (trajectories, transitions, etc.), i.e. each event is either possible or impossible.

General definitions of continuous-time Markovian systems of such kind have appeared in the literature [1]. They give a large class of (not necessarily deterministic) systems which can have both continuous and discontinuous (jump-like) trajectories. Essentially, the notion of a non-deterministic Markovian system captures the idea that only the system's current state (but not its past) determines the set of its possible futures.

Below we define the notion of a non-deterministic (complete) Markovian system in spirit of, but not exactly as in [1]. The main reasons for this are that we would like to include non-time-invariant systems in the definition and focus on partial trajectories, i.e. trajectories defined on a subset of the time scale.

We will use the following notation: $\mathbb{N} = \{1, 2, 3, \dots\}$, $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$, $f : A \rightarrow B$ is a total function from A to B , $f : A \dashrightarrow B$ is a partial function from A to B , $f|_X$ is the restriction of a function f to a set X , 2^A is the power set of a set A . The notation $f(x) \downarrow$ ($f(x) \uparrow$) means that $f(x)$ is defined (resp. undefined) on the argument x , $\text{dom}(f) = \{x \mid f(x) \downarrow\}$. Also, \neg , \vee , \wedge , \Rightarrow , \Leftrightarrow denote the logical operations of negation, disjunction, conjunction, implication and equivalence correspondingly. Let us denote:

- $T = [0, +\infty)$ is the (real) time scale. We assume that T is equipped with a topology induced by the standard topology on \mathbb{R}
- \mathfrak{T} is the set of all connected subsets of T with cardinality greater than one.

For the purpose of this paper, we will use the following definition of a dynamical system on the time scale T .

Definition 1. *A dynamical system on T is as an abstract object M (a mathematical model; in applications this may be an equation, inclusion, switched system, etc.) together with the associated time scale T (this scale will be the same*

throughout the paper), the set of states Q , and the set of (partial) trajectories Tr . A trajectory is a function $s : A \rightarrow Q$, where $A \in \mathfrak{T}$ (note that trivial trajectories defined on singleton or empty time sets are excluded). The set Tr satisfies the property: if $s : A \rightarrow Q \in Tr$, $B \in \mathfrak{T}$, and $B \subseteq A$, then $s|_B \in Tr$. We will refer to this property as "Tr is closed under proper restrictions (CPR)".

We will say that a trajectory $s_1 \in Tr$ is a *subtrajectory* of $s_2 \in Tr$ (denoted as $s_1 \sqsubseteq s_2$), if $s_1 = s_2|_A$ for some $A \in \mathfrak{T}$. The trajectories s_1 and s_2 are *incomparable*, if s_1 is not a subtrajectory of s_2 and vice versa.

According to the definition given above, for a time $t_0 \in T$ and $q_0 \in Q$ there may exist multiple incomparable trajectories s such that $s(t_0) = q_0$ (as well as one or none). In this sense a dynamical system can be non-deterministic.

It is easy to see that (Tr, \sqsubseteq) is a partially ordered set (poset).

Definition 2. A set Tr (which is CPR) is

- complete, if (Tr, \sqsubseteq) is a chain-complete poset (every chain has a supremum)
- Markovian, if $s \in Tr$ for each $s_1, s_2 \in Tr$ and $t \in T$ such that $t = \sup \text{dom}(s_1) = \inf \text{dom}(s_2)$, $s_1(t) \downarrow$, $s_2(t) \downarrow$, and $s_1(t) = s_2(t)$, where

$$s(t) = \begin{cases} s_1(t), & t \in \text{dom}(A) \\ s_2(t), & t \in \text{dom}(B) \end{cases}$$

Note that because Tr is closed under restrictions to sets $A \in \mathfrak{T}$, the supremum of a chain c in poset (Tr, \sqsubseteq) exists iff $s_* \in Tr$, where $s_* : \bigcup_{s \in c} \text{dom}(s) \rightarrow Q$ is defined as follows: $s_*(t) = s(t)$, if $s \in c$ and $t \in \text{dom}(s)$ (this definition is correct, because c is a chain with respect to subtrajectory relation).

The notions of complete and Markovian sets of trajectories are illustrated in Fig. 1 and 2.

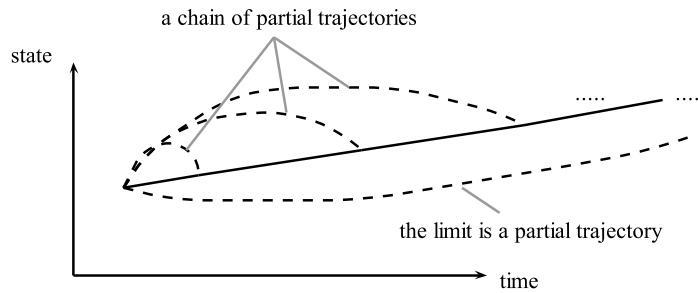


Fig. 1. Completeness property

The following proposition gives some examples of sets of trajectories.

Proposition 1. Let $Q = \mathbb{R}$. Consider the following sets of trajectories:

if one partial trajectory ends and another one begins in state q at time t (both are defined at t), then their concatenation is a partial trajectory

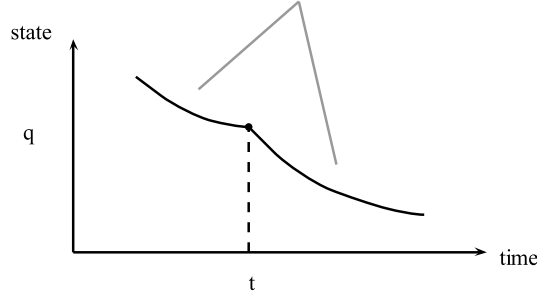


Fig. 2. Markovian property

- Tr_{all} is the set of all functions $s : A \rightarrow Q$, $A \in \mathfrak{T}$.
- Tr_{cont} is the set of all continuous functions $s \in Tr_{all}$ (on their domains)
- Tr_{diff} is the set of all differentiable functions $s \in Tr_{all}$ (on their domains)
- Tr_{bnd} is the set of all bounded functions $s \in Tr_{all}$ (on their domains).

Then the following holds:

- (1) $\emptyset, Tr_{all}, Tr_{cont}, Tr_{diff}, Tr_{bnd}, Tr_{diff} \cap Tr_{bnd}$ are CPR
- (2) $\emptyset, Tr_{all}, Tr_{cont}$ are complete and Markovian
- (3) Tr_{diff} is complete, but is not Markovian
- (4) Tr_{bnd} is Markovian, but is not complete
- (5) $Tr_{diff} \cap Tr_{bnd}$ is neither complete, nor Markovian.

Definition 3. A non-deterministic complete Markovian system (NCMS) is dynamical system is (M, T, Q, Tr) such that Tr is complete and Markovian.

The following propositions 2-4 give some examples of NCMS.

Proposition 2. Let $Q = \mathbb{R}^d$ ($d \in \mathbb{N}$) and M be a differential equation $\frac{dy}{dt} = f(t, y)$, where $f : \mathbb{R} \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a given total function. Let Tr be the set of all functions $s : A \rightarrow Q$, $A \in \mathfrak{T}$ such that s is differentiable on A and satisfies M on A . Then (M, T, Q, Tr) is a NCMS.

Proposition 3. Let M be a differential inclusion $\frac{dy}{dt} = F(t, y)$, where $F : \mathbb{R} \times \mathbb{R}^d \rightarrow 2^{\mathbb{R}^d}$ is a given (total) function. This is not necessarily a NCMS, but it can be converted to a NCMS as follows. Let M' be the system $\begin{cases} \frac{dx}{dt} = x \\ y \in F(t, x) \end{cases}$, where x is a new variable. Let $Q = \mathbb{R}^d \times \mathbb{R}^d$ and Tr be the set of all $s : A \rightarrow Q$, $A \in \mathfrak{T}$ such that s is locally absolutely continuous on A and satisfies M' almost everywhere on A (w.r.t. Lebesgue's measure). Then (M, T, Q, Tr) is a NCMS.

Proposition 4. Let Q be a set equipped with discrete topology. Let $r \subseteq Q \times Q$ be a relation on Q . Let M be a system $\begin{cases} y(t+) = y(t), & t \notin \mathbb{N}_0 \\ (y(t), y(t+)) \in r, & t \in \mathbb{N}_0 \end{cases}$, where y denotes an unknown function, $y(t+)$ denotes the right limit at t . Let Tr be the set of all piecewise-constant left-continuous functions $s : A \rightarrow Q$ (w.r.t. discrete topology on Q) which satisfy M on A (see Fig. 3). Then (M, T, Q, Tr) is a NCMS.

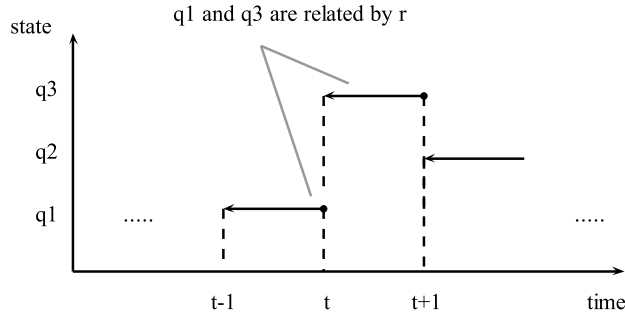


Fig. 3. A piecewise-constant left-continuous trajectory which models an execution of a (discrete-time) transition system (Q, r) .

Below we will describe a general complete Markovian set of trajectories (or a system) in terms of certain local predicates.

Let us introduce the following terminology:

Definition 4. Let $s_1, s_2 : T \rightarrow Q$. Then s_1 and s_2 :

- coincide on a set $A \subseteq T$, if $A \subseteq \text{dom}(s_1) \cap \text{dom}(s_2)$ and $s_1(t) = s_2(t)$ for each $t \in A$. We denote this as $s_1 \dot{=}_A s_2$.
- coincide in a left neighborhood of $t \in T$, if $t > 0$ and there exists $t' \in [0, t)$, such that $s_1 \dot{=}_{(t', t]} s_2$. We denote this as $s_1 \dot{=}_{t-} s_2$.
- coincide in a right neighborhood of $t \in T$, if there exists $t' > t$, such that $s_1 \dot{=}_{[t, t')} s_2$. We denote this as $s_1 \dot{=}_{t+} s_2$.

Let Q be a set of states. Denote by $ST(Q)$ the set of pairs (s, t) where $s : A \rightarrow Q$ for some $A \in \mathfrak{T}$ and $t \in A$.

Definition 5. A predicate $p : ST(Q) \rightarrow \text{Bool}$ ($\text{Bool} = \{\text{true}, \text{false}\}$) is called

- left-local, if $p(s_1, t) \Leftrightarrow p(s_2, t)$ whenever $(s_1, t), (s_2, t) \in ST(Q)$ and $s_1 \dot{=}_{t-} s_2$, and moreover, $p(s, t)$ whenever t is the least element of $\text{dom}(s)$
- right-local, if $p(s_1, t) \Leftrightarrow p(s_2, t)$ whenever $(s_1, t), (s_2, t) \in ST(Q)$, $s_1 \dot{=}_{t+} s_2$, and moreover, $p(s, t)$ whenever t is the greatest element of $\text{dom}(s)$

- left-stable, if whenever t is not the least element of $\text{dom}(s)$, $p(s, t)$ implies that there exists $t' \in [0, t)$ such that $p(s, \tau)$ for all $\tau \in [t', t] \cap \text{dom}(s)$
- right-stable, if whenever t is not the greatest element of $\text{dom}(s)$, $p(s, t)$ implies that there exists $t' > t$ such that $p(s, \tau)$ for all $\tau \in [t, t'] \cap \text{dom}(s)$.

The theorems given below show how left- and right-local predicates can be used to specify/represent a complete Markovian set of trajectories (or system).

Theorem 1. *Let $l : ST(Q) \rightarrow \text{Bool}$ be a left-local predicate and $r : ST(Q) \rightarrow \text{Bool}$ be a right-local predicate. Then the set*

$$Tr = \{s : A \rightarrow Q \mid A \in \mathfrak{T} \wedge (\forall t \in A \ l(s, t) \wedge r(s, t))\}.$$

is CPR, complete, and Markovian.

Theorem 2. *Let Tr be a CPR complete Markovian set of trajectories which take values in the set of states Q . Then there exist unique predicates $l, r : ST(Q) \rightarrow \text{Bool}$ such that l is left-local and left-stable, r is right-local and right-stable, and*

$$Tr = \{s : A \rightarrow Q \mid A \in \mathfrak{T} \wedge (\forall t \in A \ l(s, t) \wedge r(s, t))\}.$$

Let us consider an example which illustrates these theorems. Let $Q = \mathbb{R}^d$ and Tr be the set of all functions $s : A \rightarrow Q$, $A \in \mathfrak{T}$ such that s is differentiable on A and satisfies a differential equation $\frac{dy}{dt} = f(t, y)$ on A , where $f : \mathbb{R} \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ is a given function. Then Tr is complete and Markovian by Proposition 2.

Let us show how Tr can be represented using left- and right-local predicates. Let $l, r : ST(Q) \rightarrow \text{Bool}$ be predicates such that

- $l(s, t)$ iff either t is the least element of $\text{dom}(s)$, or $\partial_- s(t) \downarrow = f(t, s(t))$,
- $r(s, t)$ iff either t is the greatest element of $\text{dom}(s)$, or $\partial_+ s(t) \downarrow = f(t, s(t))$,

where $\partial_- s(t)$ and $\partial_+ s(t)$ denote the left- and right- derivative of s at t (the symbol \downarrow indicates that the left hand side of the equality is defined). It is not difficult to check that l is left-local, r is right-local, and $Tr = \{s : A \rightarrow Q \mid A \in \mathfrak{T} \wedge (\forall t \in A \ l(s, t) \wedge r(s, t))\}$. In general case, l and r are not necessarily (respectively) left- and right-stable. But we can define another predicates l_*, r_* on $ST(Q)$ such that

- $l_*(s, t)$ iff either t is the least element of $\text{dom}(s)$, or there exists $t' < t$ such that s is differentiable on $(t', t]$ and satisfies differential equation $\frac{dy}{dt} = f(t, y)$ on $(t', t]$ (at the time t the derivative is understood as left-derivative).
- $r_*(s, t)$ iff either t is the greatest element of $\text{dom}(s)$, or there exists $t' > t$ such that s is differentiable on $[t, t')$ and satisfies $\frac{dy}{dt} = f(t, y)$ on $[t, t')$.

Then it is not difficult to see that l_* is left-local and left-stable, and r_* is right-local and right-stable, and $Tr = \{s : A \rightarrow Q \mid A \in \mathfrak{T} \wedge (\forall t \in A \ l_*(s, t) \wedge r_*(s, t))\}$.

3 Existence of Global-in-Time Trajectories

Let us recall our original question about global-in-time trajectories and formulate it for non-deterministic complete Markovian systems.

Let (M, T, Q, Tr) be a NCMS. Our question (let us denote it as **Q0**) is whether it is true that for each $t_0 \in T$, $q_0 \in Q$ there exists a trajectory $s : T \rightarrow Q$ (i.e. global-in-time) such that $s(t_0) = q_0$.

Note that we ask about existence of a trajectory defined in both time directions relative to t_0 . The case when we are interested in existence of a trajectory defined in one direction (e.g $s : [t_0, +\infty) \rightarrow Q$) is not considered in this paper, but can be studied analogously.

Let us decompose **Q0** into the following two questions:

- Q1:** Is it true that for each $t_0 \in T$, $q_0 \in Q$ there exists a (partial) trajectory $s : A \rightarrow Q$ such that t_0 is an interior point of A (relative to the topology on T , e.g. 0 is considered an interior point of $[0, 1]$) and $s(t_0) = q_0$?
- Q2:** Is it true that for each partial trajectory $s : A \rightarrow Q$ such that A is a compact segment there exists a trajectory $s' : T \rightarrow Q$ such that $s = s'|_A$?

Proposition 5. *The answer to the question **Q0** is positive iff the answers to **Q1** and **Q2** are positive.*

The question **Q1** is about existence of a local-in-time trajectories. We will not study it in this paper and assume that it can be answered using domain-specific methods. Our aim is to answer **Q2** using only information about existence of locally defined trajectories in the neighborhood of each time moment.

Let us introduce several definitions. Let $\Sigma = (M, T, Q, Tr)$ be a fixed NCMS.

Definition 6. – A right dead-end path (in Σ) is a trajectory $s : A \rightarrow Q$ such that A has a form $[a, b)$, where $a, b \in T$. and there is no $s' : [a, b] \rightarrow Q \in Tr$ such that $s = s'|_{\text{dom}(s)}$ (i.e. s cannot be extended to a trajectory on $[a, b]$). The value b is called the end of this path.

- A left dead-end path (in Σ) is a trajectory $s : A \rightarrow Q$ such that A has a form $(a, b]$, where $a, b \in T$. and there is no $s' : [a, b] \rightarrow Q \in Tr$ such that $s = s'|_{\text{dom}(s)}$. The value a is called the end of this path.
- A dead-end path is either a right dead-end path, or a left dead-end path.

Let $f : [0, +\infty) \rightarrow [0, +\infty)$ be a positive-definite (i.e. $f(0) = 0$, $f(x) > 0$ when $x > 0$), monotonously non-decreasing, and continuous function (e.g. $f(x) = x$).

Definition 7. – A right dead-end path $s : [a, b) \rightarrow Q$ is called f - O_b -escapable, where O_b is a connected neighborhood of b , if there exists $c \in (a, b) \cap O_b$, $d \in [b + f(b - c), +\infty)$, and a trajectory $s' : [c, d] \cap O_b \rightarrow Q$ such that $s(c) = s'(c)$.

- A left dead-end path $s : (a, b] \rightarrow Q$ is called f - O_b -escapable, where O_b is a connected neighborhood of b , if there exists $c \in (a, b) \cap O_b$, $d \in [0, \max\{a - f(c - a), 0\}]$, and a trajectory $s' : [d, c] \cap O_b \rightarrow Q$ such that $s(c) = s'(c)$.
- A right- or left- dead-end path is called f -escapable, if it is f - T -escapable.

This definition is illustrated in Fig. 4. Note that a suffix of a right dead-end path $s : [a, b) \rightarrow Q$ (i.e. a restriction of the form $s|_{[a', b)}$, where $a' \in [a, b)$) is a right dead-end path. Analogously, a prefix of a left dead-end path $s : (a, b] \rightarrow Q$ (i.e. a restriction of the form $s|_{(a, b']}$, where $b' \in (a, b]$) is a left dead-end path.

Let $\Sigma = (M, T, Q, Tr)$ be a NCMS. For each $t \in T$ let $O_t \subseteq T$ be some connected neighborhood of t and D_t be the set of all dead-end paths s (in Σ) such that t is the end of s and $dom(s) \subseteq O_t$.

Theorem 3. *The following conditions are equivalent:*

- (1) for each partial trajectory $s : A \rightarrow Q$ such that A is a compact segment there exists a trajectory $s' : T \rightarrow Q$ such that $s = s'|_A$
- (2) each dead-end path (in Σ) is f -escapable
- (3) for each $t \in T$ and $s \in D_t$, s is f - O_t -escapable.

Note that this theorem holds for an arbitrary fixed f and arbitrary fixed choice of neighborhoods $O_t, t \in T$.

This theorem gives an answer to the question **Q2**. The condition 3 of this theorem shows in which sense Theorem 3 reduces the question of global-in-time existence of trajectories to the analysis of local existence of trajectories in the neighborhood of each time moment.

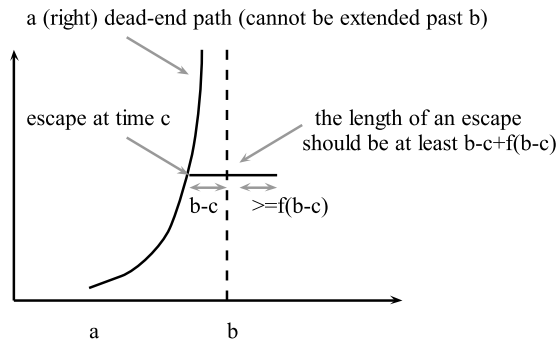


Fig. 4. An f -escapable right dead-end path.

4 Conclusion

We have studied the question of existence of global-in-time trajectories for each initial condition of a (non-time-invariant) non-deterministic complete Markovian system. We have shown that this question can be answered using analysis of existence of locally defined trajectories in a neighborhood of each time. The results can be useful for studying the problems of well-posedness and reachability for continuous and discrete-continuous (hybrid) dynamical systems.

References

1. Willems, J.: Paradigms and Puzzles in the Theory of Dynamical Systems. *IEEE Transactions on automatic control.* 36, 259–294 (1991)
2. Coddington, E., Levinson, N.: *Theory of ordinary differential equations.* McGraw-Hill, New York (1955)
3. Filippov, A.: *Differential equations with discontinuous right-hand sides.* *AMS Trans.* 42, 199–231 (1964)
4. Tangiguchi, T.: Global existence of solutions of differential inclusions. *Journal of mathematical analysis and applications.* 166, 41–51 (1992)
5. Aubin, A., Cellina, A.: *Differential inclusions.* Springer-Verlag, Berlin (1984)
6. Seah, S.W.: Existence of solutions and asymptotic equilibrium of multivalued differential systems. *J. Math. Anal. Appl.* 89, 648–663 (1982)
7. Gliklikh, Y.: Necessary and sufficient conditions for global-in-time existence of solutions of ordinary, stochastic, and parabolic differential equations. *Abstract and Applied Analysis.* 2006, 1–17 (2006)
8. Heemels, W., Camlibel, M., Van der Schaft, A.J., Schumacher, J.M.: On the existence and uniqueness of solution trajectories to hybrid dynamical systems. In: Johansson, R., Rantzer, A. (eds.) *Nonlinear and Hybrid Control in Automotive Applications.* pp. 391–422. Springer, Berlin (2003)
9. Goebel, R., Sanfelice, R., Teel, R.: Hybrid dynamical systems. *IEEE Control Systems Magazine.* 29, 29–93 (2009)
10. Henzinger, T.: The theory of hybrid automata. *IEEE Symposium on Logic in Computer Science.* 278–292 (1996)
11. Constantin, A.: Global existence of solutions for perturbed differential equations. *Annali di Matematica Pura ed Applicata.* 168, 237–299 (1995)
12. Doob, J.B.: *Stochastic processes.* Wiley-Interscience (1990)

Verification of Systems: Deadlock Analysis Based on Petri Nets

Štefan Hudák

Department of Computers and Informatics
FEI TU 04011 Košice,
Letná 9, Slovak Republic,
e-mail:stefan.hudak@tuke.sk

Abstract. The present work is devoted to the study of deadlock problem in Place/Transition (P/T) nets, particularly to the exploration of how a deadlocks' presence can be revealed solely on the basis of the P/T net N_0 in question and a structure, here termed as *the fsa of the type M_w* , that represents the reachability set $\mathfrak{R}(N_0)$ of N_0 . The structure can be obtained from N_0 following the original algorithm for solving the reachability problem RP for Petri Nets in general case by the author. It turns out, that the structure of M_w bears some important properties with respect to the deadlock analysis.

Deadlock analysis is an important part of system verification, so the results achieved can be of some value to that. It is demonstrated that results presented are quite significant, and cover some gap in both, theory and practice of the deadlock analysis of state-based systems, particularly those whose specification can be expressed via Petri Nets.

Keywords. Place/Transition Nets, deadlock analysis, reachability, finite state representation of the state reachability set, finite state automaton of the type M_w .

Key Terms. Mathematical Model, Specification Process, Verification Process

1 Introduction

In the development of (state-based) system, the design of a system in question, is the core of the process. The latter is actually a realization of the what requirement specification is about. There are two intrinsic activities of any development: *validation* and *verification*. The validation is the process of assurance that the design will produce the right system (according to requirements), while the verification assures that the design is carried on properly (according to a particular design principle)[21]. Verifying the system designed on the presence (absence) of deadlock situations (deadlock analysis-DA) might be a part of verification process. In the paper we pay attention to the deadlock analysis.

The approach to deadlock analysis applied in the paper is based on the reachability analysis [6] made on the representation (model) of the system designed in Petri Nets [8]. In [6] an original method (algorithm) to analyze and solve the reachability problem (RP) for Petri Nets in general case was introduced. The method is based on the structure, we have called it *the finite state automaton of the type M_w* , that is created by the algorithm, and the analysis of the structure based on results of automata theory and the convex analysis of the state space represented by M_w . Some authors in the field of Petri Nets used to use for representing state space of reachable states of Petri nets the structure termed as *coverability graph*. The two notions coincide to some extent, but differ in significant number of cases.

The approach is founded on the information that was neglected and suppressed by the RP algorithm while creating M_w , and thus hidden in it. The modification of M_w construction, that is introduced in this paper, discloses the information previously hidden to serve the purpose mentioned.

The paper consists of four parts. In the first part basic notions and results concerned Petri Nets, reachability and deadlock analysis are given. The second part deals with the algebraic properties of M_w . It turns out that M_w is finite state automaton, with some interpretations of its states via k -dimensional nonnegative integer ω vectors. Each such ω vector represents a state subspace of PN in question, and can be thought of as a poset. That view on ω states allows us to establish relation among deadlocks and minimal or least elements of such the posets. In the third part we deal with the issue of disclosing the information previously hidden, and define more precisely the new notion and denotation of ω coordinates. The fourth part consists of an application of the theory of ω coordinates developed. The application is made to two PNs, which was introduced by T.Murata [1] as manifestation of the fact, that using coverability graphs as the representation of state space of PN is weak and insufficient for disclosing deadlocks in PN. The same conclusion was jumped to in [2].

2 Some basic preliminaries to DA

In the paper we denote by \mathbb{N} the set of natural numbers $\{0, 1, 2, \dots\}$, by \mathbf{Z} the set of all integers, \mathbf{Z}^k (\mathbb{N}^k) the set of k -dimensional (nonnegative)integer vectors. A notion of (k -dimensional) *vector addition system* (VAS) W_k is a couple

$$W_k = (q_0, W)$$

where $q_0 \in \mathbb{N}^k$ is the initial state of W_k , W is a finite set of (k -dimensional integer) vectors. We call a reachable state vector of W_k each $q \in \mathbb{N}^k$ such that

1. $q = q_0 + w_{i_1} + \dots + w_{i_n}$ for some integer $n \geq 0$, $w_{i_j} \in W$, $j = 1, \dots, n$
and
2. for $\forall j(1 \leq j < n) : q_j = q_0 + w_{i_1} + \dots + w_{i_j} \in \mathbb{N}^k$

Here by $+$ we mean the operation of vector addition. We call the set of all such vectors *the reachability set* of VAS W_k , and denote it as $R(W_k)$. Given any VAS

$W_k = (q_0, W)$ then for any $q \in \mathbb{N}^k$ a problem whether $q \in R(W_k)$ is called the *reachability problem* of VAS (with respect to q). We will occasionally use the abbreviation $RP(q, W_k)$ for it.

With any VAS $W_k = (q_0, W)$ we can associate a tree structure, which we call the *vector state tree*, VST_w , and we mean by that a double labelled oriented rooted tree $VST_w = (T_w, Lab(V), Lab(E), q_0)$, $T_w = (V, E, r_0)$ is an oriented rooted tree, V - a set of vertices, $E \subseteq V \times V$ - a set of edges, $r_0 \in V$ - the root of T_w , $Lab(V) \subseteq \mathbb{N}^k$ - a set of vertex labels, $Lab(E) \subseteq W$ - a set of edge labels that are defined as follows: there are two labelling mappings $lab_1 : V \rightarrow Lab(V), lab_2 : E \rightarrow Lab(E)$ such that $lab_1(r_0) = q_0$ and any vertex of T_w $v \in V$ with $lab_1(v) = q$ has a son $u \in V$ with $lab_1(u) = q'$ and $lab(v, u) = a$ iff $q' = q + a$.

As a very consequence of the above definition we have that $Lab(V) = R(W_k)$ where $W_k = (q_0, W)$ is the VAS and we can alternatively write $VST_w = (T_w, R(W_k), Lab(E), q_0)$ □

2.1 Place/Transition Nets (P/T Nets).

Place/Transition (P/T) Nets stand here for a class of Petri Nets in which multiple arcs are allowed and places have unlimited capacities. For more details on PN we refer the reader to the literature, e.g. [8]. For any P/T net $N_0 = (P, T, pre, post, m_0)$, where P is a finite set of places, T is a finite set of transitions, $pre : P \times T \rightarrow \mathbb{N}$ - preset function, and $post : P \times T \rightarrow \mathbb{N}$ - postset function, that all define a structure on the set $P \cup T$. It is very common to represent the P/T Net ¹ by the oriented bipartite graph (Fig. 1).

Here we have:

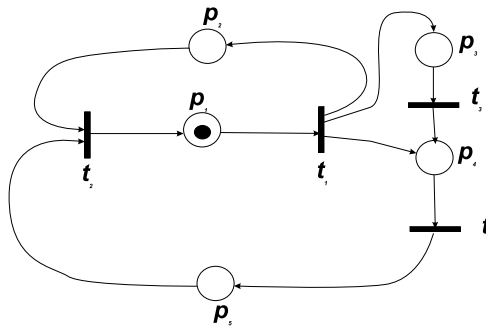


Fig. 1. Graph representation of Petri Net

¹ We will use Petri Net (PN) occasionally instead of P/T Net, so we consider them as synonyms

$$P = \{p_1, p_2, p_3, p_4, p_5\}$$

$$T = \{t_1, t_2, t_3, t_4\}$$

pre and *post* functions are given in Table 1 and Table 2 respectively.

Table 1:

P	T	pre(p,t)
p_1	t_1	1
p_2	t_2	1
p_5	t_2	1
p_3	t_3	1
p_4	t_4	1
otherwise		0

Table 2:

P	T	post(p,t)
p_1	t_2	1
p_2	t_1	1
p_3	t_1	1
p_4	t_1	1
p_4	t_3	1
p_5	t_4	1
otherwise		0

In Fig. 2 there is a correspondence shown between the graph representation of PN N and *pre* and *post* functions.

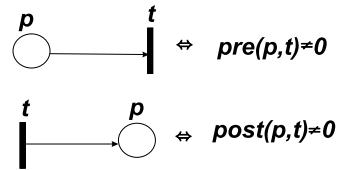


Fig. 2. The correspondence between the graph representation of PN and the *pre* and *post* functions

The following useful notations can be defined:

- $\bullet t = \{p \mid pre(p, t) \neq 0\}$ the set of preconditions of t
- $t^\bullet = \{p \mid post(p, t) \neq 0\}$ the set of postconditions of t
- $p^\bullet = \{t \mid pre(p, t) \neq 0\}$
- $\bullet p = \{t \mid post(p, t) \neq 0\}$

By the marking of PN $N = (P, T, pre, post)$ we mean a totally defined function

$$m : P \longrightarrow \mathbb{N} \tag{1}$$

$$\tag{2}$$

We use m to describe the situation or configuration in PN N . Namely we say the condition represented by the place p in PN N holds iff $m(p) \neq 0$. Without loss of generality we assume that P and T have k and m elements respectively.

i.e. $P = \{p_1, p_2, \dots, p_k\}$, $T = \{t_1, t_2, \dots, t_m\}$ and we fix some ordering of both, places and transitions from now on. Using the ordering of places we can consider m to be the k -dimensional nonnegative integer vector, i.e. $\vec{m} \in \mathbb{N}^k$. More formally

$$\vec{m} = (m(p_1), m(p_2), \dots, m(p_k))$$

and $m(p_i)$ is the value of m in p_i , $i = 1, 2, \dots, k$, according to (1). In our example (Fig. 1) $m(p_i) = 1$ iff $i = 1$, or alternatively $\vec{m} = (1, 0, 0, 0, 0)$. For the simplicity we will use the denotation m for either interpretations of the marking m when it doesn't cause any troubles. We say t is enabled in m , and denote it $m \stackrel{t}{\vdash}$, iff for every $p \in \bullet t$, $m(p) \geq pre(p, t)$. In Fig. 1 t_1 is enabled in $m = (1, 0, 0, 0, 0)$ because $\bullet t_1 = \{p_1\}$ and $m(p_1) = 1$, and $pre(p_1, t_1) = 1$. In general, given PN N , a marking m of N , several transitions from T can be enabled in m . Once the transition t is enabled it can fire. The effect of the firing t in m is the creation of a new marking m' that depends on m and t . We use a denotation

$$m \stackrel{t}{\vdash} m'$$

and m' is defined in the following way:

$$m'(p) = \begin{cases} m(p) - pre(p, t) & p \in \bullet t \setminus t^\bullet \\ m(p) + post(p, t) & p \in t^\bullet \setminus \bullet t \\ m(p) - pre(p, t) + post(p, t) & p \in \bullet t \cap t^\bullet \\ m(p) & otherwise \end{cases}$$

In PN N of Fig. 1 we can write $m = (1, 0, 0, 0, 0) \stackrel{t_1}{\vdash} m' = (0, 1, 1, 1, 0)$. Notice transitions t_3, t_4 will be enabled in m' either.

We say the sequence of transitions $\sigma = t_1 t_2 \dots t_r$ is *admissible firing sequence* in PN N , provided a sequence of markings m_0, m_1, \dots, m_r exists and such that $m_{i-1} \stackrel{t_i}{\vdash} m_i$, $i = 1, 2, \dots, r$. In that case we write $m_0 \stackrel{\sigma}{\vdash} m_r$, or simply $m_0 \stackrel{*}{\vdash} m_r$, when σ is immaterial. The marking m is to be called *the reachable marking* in N from m_0 (via σ). We fix the marking m_0 to be *the initial marking* of PN $N = (P, T, pre, post)$ and we denote it $N_0 = (N, m_0)$ or $N_0 = (P, T, pre, post, m_0)$. Given PN $N_0 = (P, T, pre, post, m_0)$ we define the set of reachable markings

$$\mathcal{R}(N_0) = \{m \mid m_0 \stackrel{\sigma}{\vdash} m, \}$$

We can define the language of PN N_0

$$L(N_0) = \{\sigma \in T^* \mid m_0 \stackrel{\sigma}{\vdash} m, \sigma \in T^*\}$$

and we call it *PN language*.

2.2 VAS and Petri Nets.

Let $N_0 = (P, T, pre, post, m_0)$ be a Petri Net with the initial marking m_0 . Recall m_0 can be represented as a k -dimensional nonnegative integer vector, i.e. $\mathbf{m}_0 \in \mathbf{Z}^k$ and $\mathbf{m}_0 = (m_0(p_1), \dots, m_0(p_k))$. Let us fix an ordering of places in P and transitions in T , i.e. $P = \{p_1, \dots, p_k\}$ and $T = \{t_1, \dots, t_m\}$.

In PN literature (e.g. [6],[8]) we have the following characterization of the marking obtained (reached) in N_0 from initial marking m_0 under firing transition sequence $\sigma \in T^*$

$$m_0 \xrightarrow{\sigma} m \Leftrightarrow \mathbf{m} = \mathbf{m}_0 + (c \cdot \Psi^T(\sigma))^T$$

and $\Psi(\sigma)$ is the Parikh mapping over the (ordered) alphabet T , and $\Psi^T(\sigma)$ stands for the transposition of the row vector $\Psi(\sigma)$.

Any transition $t \in T$ can be represented as a k -dimensional integer vector

$$\mathbf{t} = \mathbf{post}(t) - \mathbf{pre}(t)$$

and

$$\begin{aligned} \mathbf{post}(t) &= ((post(p_1, t), \dots, post(p_k, t)) \\ \mathbf{pre}(t) &= ((pre(p_1, t), \dots, pre(p_k, t)) \end{aligned}$$

It can be easily seen that

$$m_0 \xrightarrow{t} m \Leftrightarrow \mathbf{m} = \mathbf{m}_0 + \mathbf{t}$$

and we can construct for PN $N_0 = (P, T, pre, post, m_0)$ the vector addition system $W_k = (q_0, W)$ such that $q_0 = \mathbf{m}_0, W = \{\mathbf{t} | t \in T\}$, and $k = cardP$. The following result holds

Theorem 1. [6]

For any PN $N_0 = (P, T, pre, post, m_0)$ there is an vector addition system $W_k = (q_0, W)$ and such that $\mathcal{R}(W_k) = \mathcal{R}(N_0)$, and $k = cardP$.

Proof: That follows from the above construction. \square

2.3 Reachability Problem

Reachability problem for Petri nets attracted a lot of attention of experts in computer science community. It lasted pretty long time (almost 20 years) a solution to RP had been obtained [9],[10],[11],[12],[4]. A full account of the solution of RP by the author, including the complexity issue of RP- the upper bound of the worst-case time complexity of the solution, can be found in [6].

We are going now to describe shortly main steps of the author's RP solution.

1. Any P/T net $N_0 = (P, T, pre, post, m_0)$ can be assigned a vector addition system (VAS) $W_k = (q_0, W)$ via a representation of transitions of P/T net N_0 as vectors, where $W = \{\vec{t}_i \mid t_i \in T\}$, ($q_0 = m_0$) and $\vec{t}_i = (post(p_1, t_i) - pre(p_1, t_i), \dots, post(p_k, t_i) - pre(p_k, t_i))$, provided $P = \{p_1, \dots, p_k\}$. By that virtue the computations of P/T net N_0 are in 1-1 correspondence with computations of the VAS W_k and $\mathcal{R}(W_k) = \mathcal{R}(N_0)$ (see Theorem 1). The computations of the VAS W_k can be represented via rooted labelled tree, termed as *vector state tree - VST_w* , whose vertices are labelled by reachable states and edges are labelled by vectorized transitions.
2. Given VST_w and its vertex with the label q , it can be characterized by two languages: \mathbf{X}_q , \mathbf{Y}_q , prefix and suffix language respectively/ which denotes labelling of paths leading to or from the vertex with the label q . The paths on VST_w can be classified, based on the length of the paths: finite and infinite, on the one side, and also based on the finite or infinite set of vector-states: vertex labellings on the other side. Any path on VST_w , outgoing from the root vertex r_0 , labelled by q_0 , can be assigned a sequence of its vertex labels (reachable) vector-states

$$s = \{q_0, q_1, \dots, q_i, \dots\} \quad (3)$$

The states in (3) are reachable states, that are vectors, i.e. $q_i \in \mathbf{N}^k$ ($k=|P|$), so for any pair of states in (3)- (q_i, q_j) , $i < j$, we can test their comparability, w.r.t. the relation \leq defined on vectors. (of the same dimension). The sequence (3) can be accompanied by the sequence of suffix(prefix) languages associated with the states of the sequence (3) . By the nature and due to properties of VASs and their computations that is clear that

$$q_i \leq q_j \Rightarrow \mathbf{Y}_{q_i} \subseteq \mathbf{Y}_{q_j} \quad (4)$$

The necessary and sufficient conditions can be formulated for a path being infinite with finite or infinite set of reachable states. Based on that a theory of transformation of infinite paths (a graph morphism), that allows pruning infinite paths and replacing them by loop-like subgraphs and thus transforming the tree into a rooted graph (vector state graph-*vsg*). The transformation ($T_{<_A^m}$) has a significant property that suffix language of the root of the original $VST_w - \mathbf{Y}_{q_0}$ is included in the suffix language of the root of the resulting vsg $T_{<_A^m}(q_0)$, i.e. $\mathbf{Y}_{q_0} \subseteq \mathbf{Y}_{T_{<_A^m}(q_0)}$. In the case the strong inequality holds between the two states on the path ($q \leq q'$ and $q \neq q'$), that causes introducing so-called ω -coordinates, that means replacing the coordinates of the both states in which the strong inequality ($<$) holds, by the special value ω , and thus creating the ω -lized state $\omega_A q$ and $\omega_A q'$, that become identical, i.e. $\omega_A q = \omega_A q'$ (A is the set of coordinate indices on which the relation $<$ holds).

By that virtue, due to the properties of ω ($\omega + a = \omega - a = \omega$ for any natural number a), any such the transformation has two-side effect: pruning the infinite path by replacing it by a finite (loop-like) subgraph, and lowering

number of coordinates w.r.t. which a comparison satisfiability of reachable (macro) states should be checked. That guaranties that in a finite number of transformation steps a finite (rooted) vsg structure \mathcal{T}_f^ω can be obtained. The significant property of the vsg \mathcal{T}_f^ω is that $\mathbf{Y}_{T^*(q_0)} \supseteq \mathbf{Y}_{q_0}$, provided that $T^*(q_0)$ is the macrostate on which the initial state q_0 is mapped after the sequence of transformations denoted as T^* .

3. Vsg \mathcal{T}_f^ω can be thought of as a special kind of *finite state automaton (fsa)* with some interpretation of its states, and with the input alphabet $W = \{\vec{t}_i \mid t_i \in T\}$. The definition of the automaton (we used to call it *finite state automaton (fsa) (of the type) M_w*) can be given as $M_w = (Q_f, W, \delta, \rho_0)$, provided the vsg $\mathcal{T}_f^\omega = (Q_f, \mathcal{T}_f, \rho_0$ and \mathcal{T}_f is the graph representation of state transition function δ . To characterize the behaviour of fsa M_w we introduce special regular expressions (*wre*-vector regular expressions (w stands here after the set W of vectors)). Any wre α is given two semantics: $[\alpha]$ -vector semantics; $\llbracket \alpha \rrbracket$ - (ordinary) language semantics. Let \mathcal{L}^{ρ_0} be the wre that denotes the language of M_w (i.e. $L(M_w) = \llbracket \mathcal{L}^{\rho_0} \rrbracket$), and q_0 to be the initial state of VAS \mathcal{W}_k . Then $[q_0 \mathcal{L}^{\rho_0}]$ denotes all reachable states. To be more precise

$$\begin{aligned} [u] &= [u] \text{ if } u \in W \\ [au] &= a + [u] \text{ if } a \in W \text{ and } u \in W^* \\ \forall q \in \mathbf{N}^k, u \in W^* \quad [qu] &= q + [u] \text{ and } \forall i (1 \leq i \leq |u_i|) \cdot q_i = [qu_i] \in \mathbf{N}^k, \\ [q_0 \mathcal{L}^{\rho_0}] &= \left\{ q' \mid q' = [q_0 u], u \in \llbracket \mathcal{L}^{\rho_0} \rrbracket \right\} \end{aligned} \quad (5)$$

4. Having constructed fsa M_w it is worth to say few words about its structure w.r.t. how it can be useful in RP solving:
- The structure of the state diagram of M_w is, in almost all cases, consisting of $n \geq 1$ strongly connected components (scc), due to transformations applied to VST_w initially and to vsg afterwards. The class of scc-like M_w s, can be divided into two subclasses. The first subclass contains M_w s whose states are labelled by simple (k-dimensional) nonnegative integer vectors. Such M_w manifests that P/T net in question N_0 (and corresponding VAS \mathcal{W}_k) has finite set of reachable states. The second subclass consists of M_w s whose states are labelled by ω (k-dimensional) nonnegative integer vectors (vectors having at least one ω coordinate). Such M_w manifests that P/T net in question N_0 (and corresponding VAS \mathcal{W}_k) has infinite set of reachable states.
 - The way how to solve the reachability problem w.r.t. a state $q \in \mathbf{N}^k$ will differ depending on whether $\mathcal{R}(N_0)$ is finite or infinite. In the finite case $RP(q, N_0)$ can be solved trivially by inspecting the state diagram of M_w and checking whether there is a state with the label q or not. In the second case we have to do the following steps:
 - 1) to find a state ρ of M_w such that $q \leq \rho$; if such a state does not exist, then $RP(q, N_0)$ has negative solution ($q \notin \mathcal{R}(N_0)$).

- 2) assume we found such the state ρ ; now we have to construct a path leading from the root state ρ_0 that is the image of the initial state q_0 under chain of transformations ($\rho = T_{<_A^m}^*(q_0)$) (for more details see [6]). By that way a wre u over the alphabet $W_L \cup W$ (W_L is the alphabet of (ρ_0) -simple loops of scc, i.e. $W_L \subseteq W^*$) can be constructed, yielding the equation

$$[q_0uv] = q \quad (6)$$

Wre u (under assumption of one scc in M_w , rooted in ρ_0) has the structure $u = \ell_1\ell_2\dots\ell_p$, where $\ell_i \in W_L$, $i = 1, 2, \dots, p$ and v is a path leading from ρ_0 to ρ such that $q \leq \rho$.

- The equation (6) yields integer linear programming problem (ILP)

$$\begin{aligned} \mathbf{AX} &= B(q), & B(q) &= q - q_0 - [v] \\ A &= ([\ell_1]^T, [\ell_2]^T \dots [\ell_{m_0}]^T) \end{aligned} \quad (7)$$

provided $W_L = \{\ell_1, \ell_2, \dots, \ell_{m_0}\}$.

5. ILP constructed does not express exactly conditions to hold for the reachability of the state q . The reason is that at building ILP (7) based on (6) some information is lost. Particularly the information that is connected with an ordering of loops passed, that is prescribed by definition of $[q_0uv]$ (all reachable states by wre uv). To check the so called 'proper choice condition' property special test should be performed, that is expressed in the predicate $con_{W_k}(A, X_0, B_0)$. So finally the RP algorithm is

RP algorithm:

Given: VAS $W_k = (q_0, W)$, $q \in \mathbb{N}^k$ - a state to be decided reachable or not;

- Step 1 : Create fsa M_w ;
 Step 2 : Construct $MILP_w(A, X_0, B(q), r)$;
 Step 3 : if $MILP_w(A, X_0, B(q), r) = true$ then go to Step 4
 else go to Step 5 ;
 Step 4 : $q \in R(W_k)$. Stop.
 Step 5 : $q \notin R(W_k)$. Stop.

We use the abbreviation

$$MILP_{W_k}(A, X_0, B(q)) \equiv ILP_{W_k}(A, X_0, B(q)) \wedge con_{W_k}(A, X_0, B_0)$$

Finally

$$RP(q, W_k) \equiv MILP_{W_k}(A, X, B(q))$$

Since ILP is decidable, and also due to finiteness of X_0 establishing truth of $con_{W_k}(A, X_0, B_0)$ is also decidable, so is the reachability problem.

3 Algebraic properties of M_w automaton

Let us have one more look at fsa $M_w = (Q, W, \delta, \rho_0)$. For simplicity let us assume that M_w consists of single scc with its root state ρ_0 . Example of such M_w is depicted in Fig. 3

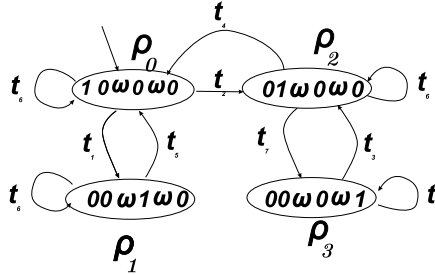


Fig. 3. State diagram of fsa M_w with a single strongly connected component

Notice that all states of the state diagram are labelled with ω vectors, e.g. $\rho_0 = (1, 0, \omega, 0, \omega, 0)$, $\rho_1 = (0, 0, \omega, 1, \omega, 0)$, $\rho_2 = (0, 1, \omega, 0, \omega, 0)$, $\rho_3 = (0, 0, \omega, 0, \omega, 1)$. Important feature of labels of the states of the M_w 's state diagram is that they are mutually incomparable as vectors.

We can look at labels of the states of the M_w 's state diagram as *macrostates*, that represent (cover) sets of reachable states. For that, we may call any macrostate ρ - a label of a state in M_w 's state diagram, as the *reachable macrostate*. We will say that two macrostates ρ and ρ' are *comparable*, and we write $\rho \leq \rho'$, provided that ρ' covers at least those reachable states, that are covered by ρ .

To express it more formally, we introduce for the macrostate ρ the set of covered reachable states- denoted by S_ρ i.e.

$$S_\rho = \{q | q \in R(N_0), q \leq \rho\}$$

S_ρ is simply *partially ordered set*(poset). The notion is well-known [19]. For any poset, particularly for S_ρ , there can be found the set of *minimal*, or *maximal* elements (states) respectively. The notions *the lower bound*, or *the upper bound* of the states of S_ρ are also well defined and used. The notion *the greatest lower bound* (glb), and *the least upper bound* (lub) are also used in that context. We only mention here, that while minimal (maximal) states belong to S_ρ , that might not be true for glb or lub respectively. We will use the notation \sqcup, \sqcap to denote the binary operation of calculating $lub(q, q') = q \sqcup q'$, or $glb(q, q') = q \sqcap q'$ respectively.

For the definition of poset, and further properties and other results reader can consult a specialized literature on the subject, e.g. see [18],[19].

For our purpose to use the information captured in fsa M_w for deadlock analysis we have to modify the algorithm of creating M_w . The information that is hidden in the state diagram of M_w is the value of particular coordinate of the state q at the moment when the coordinate is being ω -lized. .

To capture the information hidden (and lost) by the original algorithm to construct the fsa of the type M_w , we have to distinguish three types of indexing ω coordinates:

- ω_a^Δ - denotes an ω coordinate in a loop-root state ρ , with initial value of the coordinate a , with Δ as a loop added value to the coordinate after each repetition of the loop; such the ω is called the *independent root* ω ,
- $\omega_b^{\frac{i,t,\Delta_j}{}}$ - denotes so-called *dependent* ω coordinate in a loop-root state ρ , that depends on i -th ω coordinate that should generate (via repetition of its loop) a minimal value v in i -th coordinate such that $v \geq \text{pre}(p_i, t)$ for the transition t to be fireable in corresponding state while the initial value of the coordinate the dependant ω belongs to is b ; Δ_j is the increase of the *dependent* ω (in the j -th coordinate) caused by the repetition of the loop started by the transition t .
- ω_c - denotes so-called *overflowed* ω coordinate with the minimal initial value of the coordinate at the time it was overflowed for the first time.

To get a flavour why we are introducing indexed ω s, we are now turning our attention to properties of the poset $\mathbf{S}_\rho = (S_\rho, \leq, \sqcup, \sqcap)$ with respect to a deadlock state π , that can be eventually covered by a macrostate ρ , i.e. $\pi \leq \rho$.

3.1 Properties of the poset \mathbf{S}_ρ with respect to the deadlock analysis

In the previous section we have discovered, that any macrostate ρ can be taken as the poset $\mathbf{S}_\rho = (S_\rho, \leq, \sqcup, \sqcap)$. For the discovering a deadlock state of P/T net N_0 we would like to make a use of information captured in the M_w 's state diagram, specifically in macrostates by which the states are labelled with.

Assume that we are given ω -state ρ , and to be more specific, let's say that

$$\rho = \omega_A q = (\rho_1, \rho_2, \dots, \rho_k) \quad (8)$$

where $q = (q_1, q_2, \dots, q_k)$ is a reachable state, $A \subseteq \{1, 2, \dots, k\} = K$ is the set of indices in which ρ has ω - coordinates. To put it in other words that means that

$$\rho_j = \begin{cases} \varpi & \text{if } j \in A \\ q_j & \text{if } j \notin A \end{cases}$$

$$\text{and } \varpi \in \left\{ \omega_a^{\Delta_i}, \omega_b^{\frac{i,t,\Delta_j}{}}, \omega_c \right\}.$$

Now we are introducing some notions.

First we fix the macrostate ρ and its representation (8). We define

$$\text{Base}\rho = \{ q_{\rho,i}^B = (q'_1, q'_2, \dots, q'_k) \mid i \in A, q'_\ell = \rho_\ell \text{ if } i \neq \ell, \ell \in K - \{i\}, \quad (9)$$

$$q'_i = r \text{ if } \rho_i = \varpi \}$$

where $\varpi \in \{ \omega_r^{\Delta_i}, \omega_r^{j,t,\Delta_i}, \omega_r \}$.

That is clear that every $q_{\rho,i}^B \leq \rho$; in a case ρ has only one ω coordinate then $q_{\rho,i}^B$ is the $glb(\mathbf{S}_\rho)$. In the case that $\|A\| > 1$ $q_{\rho,i}^B$ is the macrostate covering a set of minimal elements of the poset \mathbf{S}_ρ .

We will call the macrostates labelling states of M_w the *reachable macrostates*. Any macrostate $\pi \leq \rho$ we will call also the *reachable macrostate*. From that point of view we may consider elements of $\text{Base}\rho$ as the collection of reachable macrostates.

Still another notion should be introduced; we define

$$q_\rho^B = (q''_1, q''_2, \dots, q''_k) \quad (10)$$

where

$$q''_{j_r} = r \Leftrightarrow \rho_j = \varpi$$

$$q''_j = \rho_j \Leftrightarrow \rho_j \in \mathbf{N}$$

and $\varpi \in \{ \omega_r^{\Delta_j}, \omega_r^{i,t,\Delta_j}, \omega_r \}$ for some $r \in \mathbf{N}$.

It is clear that

$$q_\rho^B \leq \rho$$

The crucial problem is to decide whether q_ρ^B is reachable state or not. In the latter case it will be called *spurious state* [1]. We will postpone answering that question later on.

Any *deadlock state* of P/T net

$$N_0 = (P, T, pre, post, q_0)$$

is such a state q , that is

1. reachable state, i.e. $q \in \mathcal{R}(N_0)$, and
2. for any transition $t \in T$ and at least one $p \in \bullet t$, $pre(p, t) > q(p)$

In other words there are not enough tokens at least in one of pre-places $\bullet t$ of any $t \in T$.

Assume we have a deadlock state $d \in \rho$; that means that any reachable state $q \in \rho$ and such that $q \leq d$ will be a deadlock state either. From that we have immediately, that if q_ρ^B were reachable state, it would be a deadlock state of P/T net $N_0 = (P, T, pre, post, q_0)$ since it would have been either the least or minimal element of the poset \mathbf{S}_ρ . We can summarize the properties described.

Assertion 1 Let $S_\rho = (S_\rho, \leq, \sqcup, \sqcap)$ be the poset formed by the macrostate ρ of the fsa M_w representing the set of reachable states of a P/T net $N_0 = (P, T, pre, post, q_0)$. Then if there exists a deadlock state d in ρ , then at least one of minimal elements or the least element of the poset $S_\rho - q_{min}$ will be the deadlock state too and such that $q_{min} \leq d$. \square

So, it means that the least and minimal elements of the poset S_ρ , if they exist, serve as a good indicator of presence and/or absence of deadlocks in the system represented by any P/T net.

In the algorithm of the construction of fsa M_w [6] we apply some transformations to the paths of the tree of computations of the VAS $W_k = (q_0, W)$, which results in introducing ω values into corresponding coordinates of a state vector. We have shown above there are three types of ω coordinates (ω coords in short):independent, dependent and overflowed ω coords.

The issue of creating independent ω coordinate is depicted in Fig. 4.

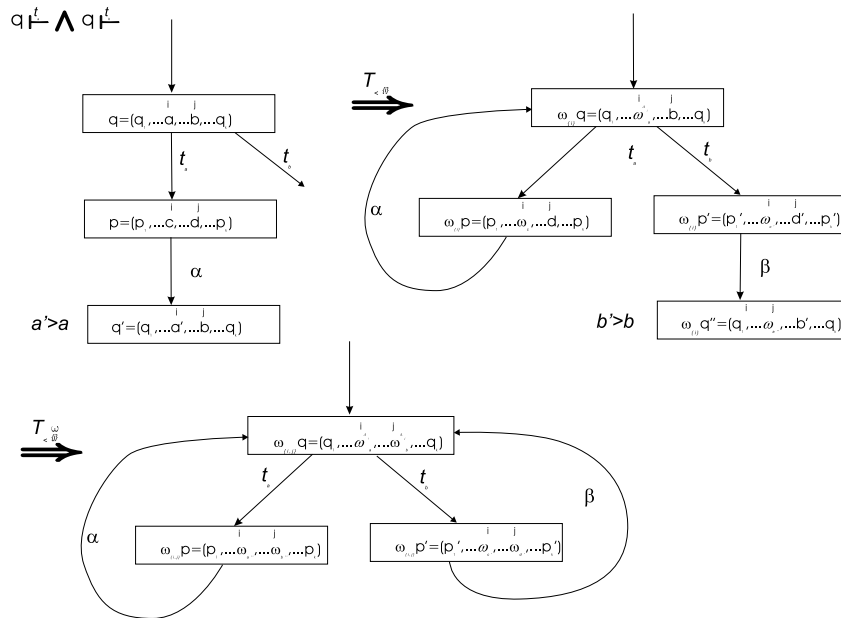


Fig. 4. Path transformation creating independent ω s

There is a state q , having values a and b in its i -th and j -th coordinate respectively. There are also two transitions (vectors) say t_a and t_b that are fireable (applicable) in q . An application of t_a at q followed by other transitions leads to a state $q' = (q_1, \dots, a', \dots, b, \dots, q_k)$, where $a < a'$ is a new value of the i -th

coordinate, and according to the algorithm which constructs automaton M_w the transformation $T_{<_i}$ applies, that creates a loop labelled with the string $\tau_a = t_a\alpha$, that replaces the path leading from q to q' which is labelled with $\tau_a = t_a\alpha$ as well. The effect of the transformation is that the state q' collapses to q creating a new state $\omega_{\{i\}}q = (q1, \dots, \omega, \dots, b, \dots, q_k)$. Because we are interested in the data keeping information on the value of the i -th coordinate which has been replaced by ω , we suggest to keep the data as a part of the new 'value' of the coordinate. Another information which we are interested in is the value $\Delta_i = a' - a$ which expresses the increase of the i -th coordinate after repetition of the loop $\tau_a = t_a\alpha$. After all we prefer to denote the new ω value of the i -th coordinate by $\omega_a^{\Delta_i}$, and to call such the ω coordinate to be *independent* ω coordinate. Another *independent* ω coordinate can be created due to firing the sequence of transitions $\tau_b = t_b\beta$ starting at the state $\omega_{\{i\}}q$. Notice that in the intermediate state p in the i -th coordinate so called overflowed ω_c cord appears. The case of creating dependent ω cords can be visualized in similar way. Due to space lack we have to skip that and we refer the reader to [7].

4 Analysis of creating ω coordinates

It was already mentioned the importance of the least and minimal states with respect to (wrt) deadlock analysis of P/T net (sect.3.1). Now we return back to the problem with an aim to analyze in the depth the issue of the role least or minimal states will play in the reachability and deadlok analysis. The main problem here is to decide in any particular case of least and/or minimal state q_ρ^B whether it is reachable or not.

Let us consider that $q_{\rho,i}^B \in Base\rho$; then $q_{\rho,i}^B \subseteq \mathfrak{R}(N_0)$ and thus $q_{\rho,i}^B$ is a macrostate covering a set of reachable states which all have the same i -th coordinate, say a_i . Moreover, the macrostate $q_{\rho,i}^B$ is the macrostate consisting of minimal states wrt macrostate ρ . The nature of the fsa of the type M_w [6] guarantees nonemptiness of $q_{\rho,i}^B$. The latter guarantees an existence of at least one reachable minimal element belonging to the macrostate $q_{\rho,i}^B$.

The state q_ρ^B can be considered to be the least element of the poset S_ρ , provided it is a reachable state, otherwise it can serve as a lower bound for the reachable states- elements of S_ρ . Very often such the state q_ρ^B will be just *spurious* reachable state, and there will be a need for q_ρ^B to be proved its reachability. To underpin that assertion some kind of analysis should be introduced first.

The case analysis of different types of $n > 1$ ω coordinates have been accomplished [7]. In every case there that has been proven, that either $Base\rho \subseteq \mathfrak{R}(N_0)$, or $q_\rho^B \in \mathfrak{R}(N_0)$. The case of $n=2$ ω coordinates is quite simple. The case of $n \geq 3$ is more complicated. There is few typical situations in the case of $n=3$ ω coordinates that shows the table below.

In the table the entry $(1, 1 \rightarrow, 1)$ stands for a dependence of ω cord *dep* on *ow*, and the entry $(1, \leftarrow 1, 1)$ stands for a dependence of ω cord *dep* on *ind*. We illustrate that only for two cases:(3,0,0).

Table:Case analysis for 3 ω cords			
Type of ω cords			
N^0	ind	dep	ow
1	3	0	0
2	2	1	0
3	2	0	1
4	1	0	2
5	1	1 \rightarrow	1
6	1	\leftarrow 1	1
7	1	\leftarrow 2	0
8	0	2 \rightarrow	1
9	0	1 \rightarrow	2
10	0	0	3

Table 1. Case analysis for 3 ω cords

The results on creating ω coordinates for the case $||A|| \leq 3$ can be generalized to any $n \in \mathbf{N}$. The conclusion we have come to is that ω coordinates can assume one of the following forms.

a) **single indices**

ind ω cord Bdep ω cord ow ω cord

$$\omega_a^{\Delta_i} \quad \omega_b^{\frac{B, t_b, \Delta_j}{B \subseteq K}} \quad \omega_c$$

b) **multiple indices**

ind ω cord

$$\omega_a^{\Delta_i} \quad \omega_{a, a'}^{\Delta_i, -} \quad \omega_{a, a', a''}^{\Delta_i, -, -}$$

Bdep ω cord

$$\omega_b^{\frac{B, t_b, \Delta_j}{B \subseteq K}} \quad \omega_{b, b'}^{\frac{B, t_b, \Delta_j, -}{b'}} \quad \omega_{b, b', b''}^{\frac{B, t_b, \Delta_j, -, -}{b', b''}}$$

ow ω cord

$$\omega_c \quad \omega_{c, c'} \quad \omega_{c, c', c''}$$

We propose to use a generalized form to represent ω coordinates, and we will call it as form (f).

$$\omega \left[\begin{pmatrix} h_1, \dots, h_k \\ d_1, \dots, d_k \end{pmatrix} \right] \quad (11)$$

where

$$\begin{aligned} \begin{pmatrix} h_1 \\ d_1 \end{pmatrix} &\in \left\{ \begin{pmatrix} \Delta_i \\ a \end{pmatrix}, \begin{pmatrix} A, t, \Delta_i \\ b \end{pmatrix} \right\} \\ \begin{pmatrix} h_i \\ d_i \end{pmatrix}_{i>1} &\in \left\{ \begin{pmatrix} - \\ a \end{pmatrix}, \begin{pmatrix} \lambda \\ c \end{pmatrix} \right\} \end{aligned} \quad (12)$$

Here λ stands for *empty* symbol. Basically, in the case of overflowed ω coordinates we will use just ω_c instead of ω_c^λ or ω_c^- . In the case of independent and dependent ω coordinates we will use instead of empty symbol '-' to visualize the correspondence of high and low indices.

In our consideration we will use shorthand notation for indexed ω coordinates as

$$\omega [I_k] \text{ where } I_k = \begin{pmatrix} h_1, \dots, h_k \\ d_1, \dots, d_k \end{pmatrix}$$

The following results have been proven as far as the generalization of the process of indexed ω coordinates is concerned:

1. the form (f) of ω coordinates has been chosen correctly, and it will be preserved by any application of $T_{< \omega}^A$ transformation, and
2. the procedure to obtain the set of minimal elements of the poset represented by a macrostate ρ -Base ρ , is determined by a choice of proper combination of low indices.

5 A case study and further analysis of ω coordinates

In his paper [1] T.Murata studied two PNs (Fig. 5) with respect to discovering liveness or deadlock, based on the coverability graphs of Petri Nets, the structure that is widely used to represent the state space of their reachable states. He showed that two PNs having the identical coverability graphs differ what concerns of liveness or deadlock properties. In this section we will use our method based on the finite automaton M_w and the properties of ω coordinates to demonstrate the power of the approach to discover safely the deadlock of Petri nets in general and it will be demonstrated by the example Petri nets by T. Murata.

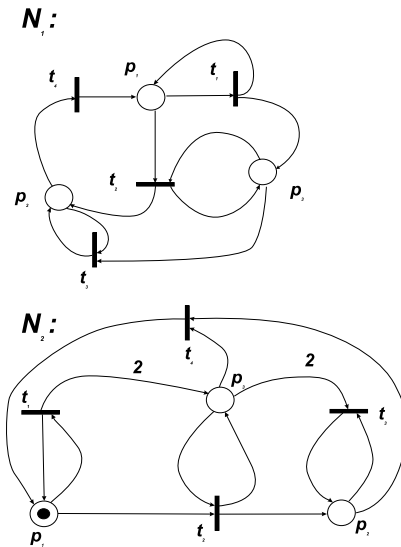


Fig. 5. Case study: Two Petri Nets with identical coverability graphs

Let us have a closer look at the two PN's. Comparing coverability graphs of the two PN's we can see they are indeed identical. Now we are going to apply the approach based on the methodology developed, that is backed by our algorithm of constructing fsa of the type M_w (in some cases state diagrams of M_w and coverability graph coincide, but in some cases they look quite different). Construction of fsas of the type M_w for the two nets can be seen in Fig. 6 and Fig. 7.

We can see that M_w automata are isomorphic, but they differ in ω cords as far as their indices are concerned.

Let us have a closer look at the M_w automata from that perspective. In M_w automaton of PN N_1 we have two macrostates: $\rho_1 = (1, 0, \omega_0^1)$ and $\rho_2 = (0, 1, \omega_{0,1})$. If we look at $\rho_1 = (1, 0, \omega_0^1)$ as at the poset, we can have the only minimal and thus the least (infimum) state

$$q_{\rho_1}^B = (1, 0, 0)$$

In the case of $\rho_2 = (0, 1, \omega_{0,1})$ we have the basis of this poset

$$Base(\rho_2) = \{(0, 1, 0), (0, 1, 1)\}$$

There are actually 2 minimal states.

Let us turn our attention to the net N_2 . In M_w automaton of PN N_2 we have also two macrostates: $\rho_1 = (1, 0, \omega_{0,1}^{2,-})$ and $\rho_2 = (0, 1, \omega_{0,2})$. If we look at ρ_1 as at the poset, we can have here no infimum state, but we still have two minimal states that creates :

$$Base(\rho_1) = \{(1, 0, 0), (1, 0, 1)\}$$

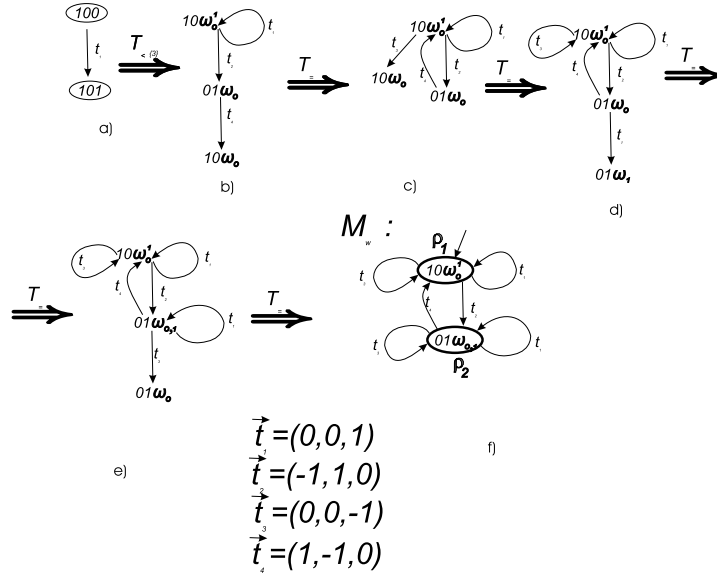


Fig. 6. M_w construction for the live Petri Net N_1 with indexed ω cords

If we look at ρ_2 as at the poset, we can have also here no infimum state, but we still have two minimal states that creates :

$$Base(\rho_2) = \{(0, 1, 0), (0, 1, 2)\}$$

5.1 Deadlock analysis

In [6] the deadlock problem is dealt with, based on the use of original RP algorithm. We wil use of the notion 'deadlock candidates' states introduced there. The latter can be derived from the structure of PN in question.

Let us consider PN $N = (P, T, pre, post)$ and let $\vec{t} = -pre(p, t) + post(p, t)$. For any $t \in T$ and $p \in P$ we say

$$p \text{ covers } t \Leftrightarrow_{df} pre(p, t) \neq 0 \tag{13}$$

In other words we are saying by (13) that

$$p \text{ covers } t \Leftrightarrow_{df} t \in p^\bullet \tag{14}$$

The (13) and (14) simply mean that p is included in t 's enabling. That is reasonable to define

$$C(p) = \{t \in T | p \text{ covers } t\}$$

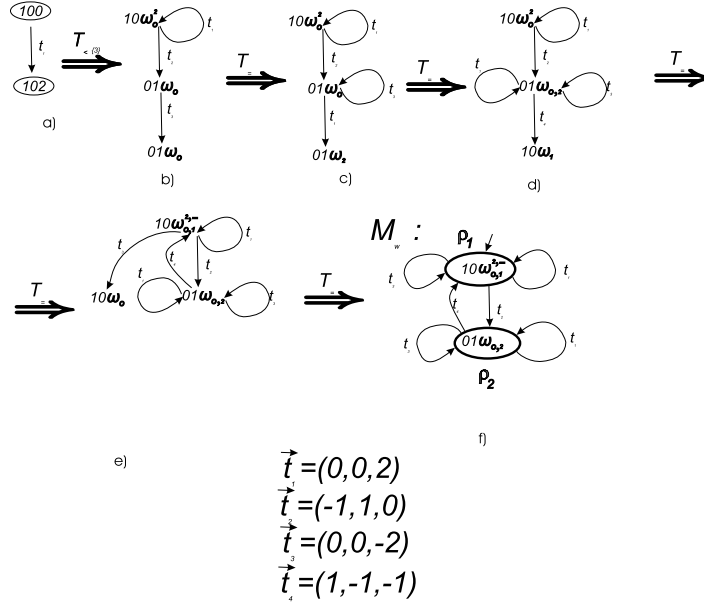


Fig. 7. M_w construction for the deadlock Petri Net N_2 with indexed ω cords

Obviously $C(p) = p^\bullet$. We are now looking for such a minimal subset C_i of P , that the union of the covers of places form the subset that will give the whole set T . We propose to call such the subset C_i the *minimal cover of T* . The notion minimal is connected with the number of places covering the set T . Notice, there can be more than one minimal cover of T .

We associate with each total cover of $T - C_i$, the set of deadlock markings, denoted as - $CanM_i$.

$$CanM_i = \{ m \in \mathbb{N}^k \mid m \leq_{C_i} m_i, p \in C_i \Rightarrow m_i(p) \leq r - 1, \\ r = \min_{r_i} \{ r_i \mid pre(p, t_i) = r_i, t_i \in p^\bullet \} \}$$

where $m \leq_{C_i} m_i \Leftrightarrow_{df} \forall p \in C_i : m(p) \leq m_i(p)$.

We can define the notion

$$CovT = \{ C_i \mid C_i \subseteq P, C_i \text{ is total cover of } T \} \quad (15)$$

Based on the $CovT$ we may define now the overall set of dead markings

$$CanM = \left\{ m \in \mathbb{N}^k \mid \exists C_i \in CovT : m \leq_{C_i} m_i, p \in C_i \Rightarrow m_i(p) \leq r - 1, \right. \\ \left. r = \min_{r_i} \{ r_i \mid pre(p, t_i) = r_i, t_i \in p^\bullet \} \right\} \quad (16)$$

Notice that notions $CanM_i$ and $CanM$ are based only on the structure of PN in question and nothing is known about the reachability of the states contained there. That is why we have used to call them 'deadlock candidates', or *potential* deadlock states. Any of such the states becomes real deadlock provided it is reachable, the issue connected with dynamic aspect of the PN in question. Now we are prepared to formalize the procedure, based on the method developed so far, as far as deadlock analysis is concerned. We do it in the form of a procedure.

DA(N_0):Algorithm for doing the deadlock analysis of Nets (P/T nets).

Input:

Petri Net (P/T net) $N_0 = (P, T.pre, post, m_0)$, of the type M_w of PN N_0

Output:

yes, if $D(N_0) \neq \Phi$
no, if $D(N_0) = \Phi$

Method:

Method is based on the results achieved in the analysis of nature of ω coordinates, that occur in ω macrostates $\rho = \omega_Aq$. Approach is based on interpretation of any such ω macrostate $\rho = \omega_Aq$ as a representation of the poset of reachable states in PN N_0 , having minimal or the least elements (MoL states). The special procedure $CanM(N_0)$ is used for creation of the set of potential deadlocks of PN N_0 .

Body of the algorithm:

```

begin
   $D(N_0) \leftarrow \Phi$ ;    */ $D(N_0)$  - the set of MoL deadlock states/*
   $D \leftarrow \Phi$ ;      */ $D$  - variable - buffer for the actual set of
MoL deadlock states/*
   $C \leftarrow CanM(N_0)$ ;    */ $D$  - variable - container of
potential deadlock states/*
   $S \leftarrow Q^\omega$ ;    */ $Q^\omega$  - the set of states of fsa  $M_w = (Q^\omega, W, \delta, \rho_0)$ 
W-the set of vectorized transitions, $\delta$ -transition function, $\rho_0$  containing  $m_0$ /*
  while  $S \neq \Phi$  do;
    begin
      choose  $\rho \in S$ ;
       $MoL \leftarrow Base\rho$ ;
       $S \leftarrow S - \{\rho\}$ ;
      if  $C \cap MoL \neq \Phi$  then  $D \leftarrow D \cup C \cap MoL$ ;
    end
    if  $D = \Phi$  then return NO
      else  $D(N_0) \leftarrow D$ ;
    return YES: $D(N_0)$ 
end

```

The algorithm $DA(N_0)$ guarantees all MoL deadlocks will be found out and delivered as the set $D(N_0)$. Actually that can be considered as solving the problem of discovering presence or absence of deadlock states in the PN N_0 .

5.2 Case study continued

We can now continue to analyze state diagrams of the two PNs. We will use the results of previous section and particularly the result of the Lemma ??.

So according to that we have to calculate now total covers for the two PNs.

In the tables below there are calculated both: the minimal total cover of T and pre-set for the net N_1 .

Total Cover of T for PN N_1					Function pre for PN N_1				
	t_1	t_2	t_3	t_4	Total Cover of T	pre	p_1	p_2	p_3
p_1	∨	∨			$C_1 = \{p_1, p_2\}$	t_1	1		
p_2			∨	∨		t_2	1		
p_3		∨	∨			t_3		1	
						t_4		1	

$$CanM = CanM_1 = \{0, 0, \omega\}$$

$$inf(1, 0, \omega_0^1) = (1, 0, 0) \notin CanM$$

$$inf(0, 1, \omega_{0,1}) \text{ nejstvuje}$$

$$Base((0, 1, \omega_{0,1})) = \{(0, 1, 0), (0, 1, 1)\} \cap CanM = \{(0, 0, \omega)\} = \Phi$$

So we jump to the conclusion that PN N_1 does not contain any deadlock!

Now we are going to turn our attention to the PN N_2 . First we calculate minimal total cover of PN N_2 and pre-set for the PN N_2 .

Total Cover of T for PN N_2					Function pre for PN N_2				
	t_1	t_2	t_3	t_4	Total Cover of T	pre	p_1	p_2	p_3
p_1	∨	∨			$C_1 = \{p_1, p_2\}$	t_1	1		
p_2			∨	∨	$C_2 = \{p_1, p_3\}$	t_2	1		1
p_3		∨	∨	∨		t_3		1	2
						t_4		1	1

$$CanM_1 = \{(0, 0, \omega)\}, CanM_2 = \{(0, \omega, 0)\}$$

$$CanM = \{(0, 0, \omega), (0, \omega, 0)\}$$

$$Base((1, 0, \omega_{0,1}^2, -)) = \{(1, 0, 0), (1, 0, 1)\}$$

$$Base((0, 1, \omega_{0,2})) = \{(0, 1, 0), (0, 1, 2)\}$$

$$Base((1, 0, \omega_{0,1}^2, -)) \cap CanM = \Phi$$

$$Base(0, 1, \omega_{0,2}) \cap CanM = \{(0, 1, 0), (0, 1, 2)\} \cap \{(0, 0, \omega), (0, \omega, 0)\} = \{(0, 1, 0)\}$$

So we may jump to the conclusion that PN N_2 has indeed deadlock state $\{(0, 1, 0)\}$!

6 Conclusion

The issue of deadlock analysis is important for the development of discrete state-based systems. The method of discovering a presence, or an absence of deadlocks in the system coined and demonstrated in the paper is based on the study of the properties of the automaton M_w . We should mention that the results presented in the paper manifest the depth and the vitality of the new method to deal with the issue of reachability in Petri Nets, particularly the part which was connected with the study of the algebraic properties of interpretations of the automaton of the type M_w . In [6] there are some results presented on the nature of that interpretation. The automaton M_w bears some similarity with coverability graphs used in Petri Nets, but as it was proven, it is more powerful to deal with deadlock analysis. Beside of that, the structure of the automaton M_w plays the central role in reachability analysis of the systems (represented via PN) with infinite state space [6]. The most important property of M_w is its reusability for reachability analysis of the PN in question wrt to any other state, not only wrt to that it was constructed for initially. On the other side it turns out that one automaton of the type M_w , say \mathcal{M} can serve in that role for whole class of PNs with the same number of places and some structure that induces corresponding set of transitions which are consistent with the \mathcal{M} structure. There is still another way how the M_w structure can be used. The fsa \mathcal{M} can be thought of as a couple $\mathcal{M}=(M,\mathcal{I})$, where M and \mathcal{I} stand for basic fsa without interpreted states and interpretation respectively. For any $k \in \mathbb{N}$ - the number of places and given structure of basic fsa M we can construct corresponding interpretation \mathcal{I} consistent with M . By that virtue, the same applies wrt doing deadlock analysis.

Due to space we have not dealt with the issue of modification of the algorithm of \mathcal{M} construction, and also many details and proofs have been skipped. There can be found in [7]. At the workplace of the author there has been environment - termed as mFDTE [16], developed. The results will be implemented in the environment. The latter combines three formal descriptions of systems: Petri Nets, process algebra, and B AMN. The latter substantiate the acronym mFDTE-multi Formal Description Techniques Environment. More details can be found in [16].

Acknowledgments. This work was supported by the Slovak Research and Development Agency grant on Modelling, simulation and implementation of GPGPU-enabled architectures of high-throughput network security tools, under the contract No. APVV-0008-10. Further it was also partially supported by the grants No. 1/3140/06 of the VEGA-The Scientific Grant Agency of Slovakia, NATO CLG982698 grant, APVV grants No. APVV-0073-07, and SK-UA-0026-07.

References

1. Murata, T.: Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE. 77 (1989)

2. Xinning Y., JiantaoZ., and Xiaoyu S.: On Reachability Graphs of Petri Nets. *Computers & Electrical Engineering*. 29, 263–272 (2003)
3. Hudák, Š.: Extensions to Petri Nets. Habilitation Thesis (in Slovak), TU Košice, 107 p. (1980)
4. Hudák, Š.: The Recursive Decidability of the Reachability Problem for Vector Addition Systems. ASMM/84, Computing Laboratory, The University of Newcastle upon Tyne, England, 78 p. (1981)
5. Hudák, Š.: On the Reachability Problem for Vector Addition Systems. Proceedings of The Second European Workshop on Application and Theory of Petri Nets, Bad Honnef, Germany, 38 p. (1981)
6. Hudák, Š.: Reachability Analysis of Systems Based on Petri Nets. elfa s.r.o. Košice, ISBN 80-88964-07-5, 272 p. (1999)
7. Hudák, Š.: Deadlock Analysis of Place/Transition Nets. Manuscript, DCI FEI TU Kosice, 79 p. (2010)
8. Reisig, W.: *Petri Nets: An Introduction*. Springer Verlag, Heidelberg (1985)
9. Sacerdote, G.S., Tenney, L.: The Decidability of the Reachability Problem for Vector Addition Systems (preliminary version). Proceedings of the 9th Ann. ACM Symposium on Theory of Computing, New York (1977)
10. Muller, H.: On the Reachability Problem for Persistent Vector Replacement Systems. *Computing Suppl.* 3, 89–104. In: Knodel, H., Schneider, H.J. (eds.) *Parallel Processes and Related Automata* (1981)
11. Mayr, E.W.: An Algorithm for the General Reachability Problem in Petri Nets. *SIAM J. of Computing*. 13 (1984)
12. Kosaraju, S.R.: Decidability of Reachability in Vector Addition Systems. In Proc. 14th Annual ACM STOC, 267–281 (1982)
13. Valk, R.: Generalizations of Petri Nets. In: Gruska, J., Chytil, M. (eds.) *MFCSS 1981*. LNCS vol.118, pp. 140–155 (1981)
14. Karp, R.M., Miller, R.E.: Parallel Program Schemata. *Journal of Computer and System Sciences*. 3, 147–195 (1969)
15. Keller, R.M.: Vector Replacement Systems: A formalism for Modeling Asynchronous Problems. Technical Report 117, Princeton University (1972)
16. Hudák, Š. et al.: A Support Tool for the Reachability and Other Petri Nets- Related Problems and Formal Design and Analysis of Discrete Systems. *Problems in Programming*. 20, 613-621, ISSN 1727-4907 (2008)
17. Hudák, Š.: De/compositional Reachability Analysis. *Journal of Electrical Engineering*, CS ISSN 0013-578X. 45, 424-431 (1994)
18. Stone, H.S.: *Discrete Mathematical Structures and Their Applications*. Science Research Associates Inc., Chicago, 402pp. (1973)
19. Best, E.: *Semantics of Sequential and Parallel Programs*. Prentice Hall Europe, ISBN 0-13-460643-4, 352pp. (1996)
20. Aho, A.V., Hopcroft, J.E., Ullman, J.D.: *Formal Languages and Their Relations to Automata*. Addison Wesley, Reading, Mass (1969)
21. Lano, K.: *The B Language and Method. A Guide to Practical Formal Development*. Springer-Verlag London, ISBN 3-540-76033-4, 232pp (1996)

Decomposition and Isomorphism of Logical Systems

Ján Bača¹

¹Department of Computers and Informatics
Technical University of Košice
Letná 9, 042 00 Košice, Slovakia
Jan.Baca@tuke.sk

Abstract. The contribution deals with decomposition of logical systems for the purpose of solving analysis, synthesis and diagnostics tasks. The system can be specified by its structure or by algebraic expressions of its function. Particular attention is paid to propose algorithms for ordering of components of algebraic expression, decomposition of algebraic expression into substrings, and composition of modularly-organized logical circuit from those substrings. Also a way for the determination of identical and isomorphic modules of de/composed circuits is presented.

Keywords. logical system, decomposition of systems, isomorphic modules, algebraic expression.

Key Terms. Mathematical Model, Research, Development.

1 Introduction

The tasks complexity of the logical systems synthesis, analysis and diagnostics are often very high and it depends on the system dimension.

If complexity $c(n)$ of the solution of the task is not linear $c(n) \neq O(n)$, but polynomial $c(n) = O(n^k)$, or exponential $c(n) = O(g^n)$, then the reduction of the total solution complexity by the decomposition of the system $S(n)$ into p subsystems $S_1(n_1), S_2(n_2), \dots, S_p(n_p)$, $n_i < n$ is used with great advantage (n – parameter which determines the system size). This is the reason for decomposition of systems into several smaller modules and solving the tasks for separate modules and composing module results back into result for whole system. Decomposition makes sense if decreasing of the solution time $t(S(n))$ in decomposed systems is bigger than increase of the solution time t_{ds} related with decomposition and back composition of the system

$$t(S(n)) > \sum_{i=1}^p t(S_i(n_i)) + t_{ds}$$

Time t_{ds} grows with numbers of modules and numbers of connections among them. The decomposition on modules with one (or minimal number of) output is more suitable. If some modules are isomorphic (or identical) it is enough solve tasks for one

of them and apply for others. For example if we generate tests for complex system, we can decompose it into several smaller modules, prepare tests for each group of isomorphic modules and compose the test for whole system on the base of module tests.

In separate tasks the systems can be described by different descriptions or formal specifications [1], [2], [3]. The definitions of the system can be split into two basic groups. In the first group the systems are defined by its function. For example the combinational logical circuit can be present by algebraic expressions of system output functions. In the second one the systems are defined by its structure. In these cases the system can be presented by netlist. The function of the system described by structure can be derived from functions of individual components and connections between them. To understand the dependence between these different system descriptions it is essential to mention, that the given function can be realized by different structures, however the given structure of the circuit realizes a unique determined function.

The logic circuit structure is given by the equation $S = (E, C)$, where E is the set of logic elements and C is the set of mutually connections, which are performed in input N_i , internal N_r or output N_o nodes.

Each element $E_j = (F_j, X_j, Y_j)$ is characterized by the logic function F_j , the inputs $X_j = (x_{j_1}, x_{j_2}, \dots, x_{j_{m_i}})$ and the outputs $Y_j = (y_{j_1}, y_{j_2}, \dots, y_{j_{m_o}})$ - $F_j : X_j \rightarrow Y_j$.

The $x_{j_u} \in N_i \cup N_r$, $y_{j_v} \in N_r \cup N_o$ signify the node names, which are the elements of the set C .

Elements $E_p = (F_p, X_p, Y_p)$, $E_q = (F_q, X_q, Y_q)$ are mutual connected, if $\{x_{p_1}, x_{p_2}, \dots, x_{p_{k_i}}, y_{p_1}, y_{p_2}, \dots, y_{p_{m_i}}\} \cap \{x_{q_1}, x_{q_2}, \dots, x_{q_{k_j}}, y_{q_1}, y_{q_2}, \dots, y_{q_{m_j}}\} \neq \emptyset$.

The module structure $S_m = (E_m, C_m)$ is connective, if for $\forall E_p, E_q \in E$, \exists the string $E_p, E_{i_1}, E_{i_2}, \dots, E_{i_u}, E_q$ ($u \geq 0$), in which all adjacent elements are mutual connected.

Elements $E_p = (F_p, X_p, Y_p)$, $E_q = (F_q, X_q, Y_q)$ are identical ($E_p \equiv E_q$), if $F_p \equiv F_q$, $X_p \equiv X_q$, $Y_p \equiv Y_q$.

Elements $E_p = (F_p, X_p, Y_p)$, $E_q = (F_q, X_q, Y_q)$ are isomorphic ($E_p \cong E_q$), if there exists such one to one correspondence $\{x_{p_1}, x_{p_2}, \dots, x_{p_{k_i}}\} \leftrightarrow \{x_{q_1}, x_{q_2}, \dots, x_{q_{k_j}}\}$ $\{y_{p_1}, y_{p_2}, \dots, y_{p_{m_i}}\} \leftrightarrow \{y_{q_1}, y_{q_2}, \dots, y_{q_{m_j}}\}$ and $F_p \equiv F_q$.

Modules $S_{m_i} = (E_{m_i}, C_{m_i})$, $S_{m_j} = (E_{m_j}, C_{m_j})$ are isomorphic, if there exists such one to one correspondence $E_{m_i} \leftrightarrow E_{m_j}$, $C_{m_i} \leftrightarrow C_{m_j}$, that $(E_p \cong E_q)$, $k = 1, 2, \dots, p$, where p is the number of module elements.

Modules $S_{m_i} = (E_{m_i}, C_{m_i})$, $S_{m_j} = (E_{m_j}, C_{m_j})$ are identical, if they are isomorphic and $C_{m_i} \equiv C_{m_j}$.

The decomposition requirements:

- connectivity of modules
- minimal number of connections to other modules
- minimal number of modules
- minimal number of module types

e) minimal number of module elements

Contradictory requirement c) and e) can be solved by hierarchy of module decomposition.

Specification of logical system by structure is more suitable for manual decomposition because in their scheme we can see structure of modules and their connection to other ones. Specification of logical system by algebraic expression of function is more suitable for automatic decomposition because in the algebraic expression we can find identical or isomorphic parts which correspond with system modules.

By decomposition of logical systems it is necessary to differentiate between combinational part and sequential part of logical circuits. Combinational part which represents excitation and output function can be described by Boolean functions. Sequential parts can be described by the transition matrix of elementary memory elements. In sequential logical circuits are usually used one type - D, T, RS or JK of elementary memory elements which present isomorphic modules with different inputs and outputs. Excitation and output function in combinational part usually are different and it is useful decompose them into several smaller modules and look for their isomorphism.

2 Algebraic Expression of a Logical System Function

In case of combinational circuits the algebraic expressions – logical operations of conjunction (AND) $x * y$, disjunction (OR) $x + y$, negation (NOT) $\sim x$ Sheffer's operations (NAND) $x | y$, Pierce's operations (NOR) $x \downarrow y$, non-equivalence (XOR) $x \oplus y$ - is fully sufficient for system specification. The brackets are used to specify priority of operations. Algebraic expressions are represented in the grammar-defined language [3]. Operation among primary inputs will be denoted as operation on level one. Operation is on level $i+1$ if maximal level of embedded operations is equal i . The maximal number of embedded operations, which must be realized by function calculation, will be denoted as depth of function embedding (the degree of the corresponding circuit).

3 De/composition of Logical Systems

3.1 Ordering of Algebraic Expression Elements

For decreasing of modules identity detection complexity, it is desirable to order the algebraic expression at first [4]. We will begin with the algebraic expression, where the individual variables are denoted as x_i , and may occur either in direct or in inverse instances in the expression. Position of the given variable in the sequence, according to which the expression will be ordered, is chosen in following manner:

$$j = 2i, \text{ for } x_i,$$

$$j = 2i - 1, \text{ for } \sim x_i, i = 1, \dots, \nu$$

where v is the number of primary inputs of the circuit. Further sequence of primary variables is obtained this way is $\sim x_1, x_1, \sim x_2, x_2, \dots, \sim x_n, x_n$,

The ordering itself is done in the bottom-up manner for every string, which represents one logical operation, separately. Composite strings, which represent the outputs of operations on the lower depth of function embedding, will be ordered by leading variables in the string, symbols of operators will be ordered by sequence: \sim (NOT), $*$ (AND), $+$ (OR), $|$ (NAND), \downarrow (NOR), \oplus (XOR). This way of the strings ordering will be denoted lexicographic ordering. The algorithm of the ordering is following:

INPUT:

An algebraic expression representation in the grammar-defined language [3].

OUTPUT:

Ordered algebraic expression.

BODY:

1. Let $i = 1$, let n be the depth of function embedding.
2. For every logical operation on the level i order the strings according lexicographic ordering.
3. $i = i + 1$
4. Repeat steps 2 and 3, until $i \leq n$.

The example of ordered algebraic expression for a circuit realizing the full adder circuit (Fig. 1) can be following. The circuit has two outputs – sum and carry into the higher level:

$$s = \sim(\sim x_1 * x_2 + x_1 * \sim x_2) * x_3 + (\sim x_1 * x_2 + x_1 * \sim x_2) * \sim x_3$$

$$c = (\sim x_1 * x_2 + x_1 * \sim x_2) * x_3 + x_1 * x_2$$

3.2 Decomposition of Algebraic Expression into Substrings

Decomposition of an algebraic expression comes out from the expression described in section 3.1, which is analyzed in bottom-up way. The result of decomposition is the decomposition of an algebraic expression into substrings that are realizable with only one elementary logical operation.

The algorithm of algebraic expression decomposition into substrings is following:

INPUT:

Ordered algebraic expression (section 3.1).

OUTPUT:

A system of algebraic expression substrings that contain exactly one logical operation.

BODY:

1. Search a substring that contains exactly one logical operation. Considered logical operations are \sim (NOT), $*$ (AND), $+$ (OR), $|$ (NAND), \downarrow (NOR), \oplus (XOR).
2. Substitute identified substring by one variable marked by symbol x and two numeric values. The first value indicates the level of substring (the depth of related function embedding). The second value identifies selected substring within the given level.

3. Compare expressions $x[i,j]$, for $j=1, 2, \dots, u$, where u is the number of expression with the level i . Identical expressions $x[i,j]$ and $x[i,j+v]$ replace by expression $x[i,j]$.
4. Express initial algebraic expression using substitution variables.
5. Repeat steps 1, 2, 3 and 4 until initial expression is reduced to logical expression containing only one logical operation.

In the case of circuit with more outputs it is necessary to execute this algorithm as well as the next algorithm for algebraic expressions of all circuit output functions.

Example of an application of the presented algorithm for the circuit realizing full adder circuit is presented in the Table 1.

Table 1. Decomposition of full adder algebraic expression into substrings

Function	Substitution	Identity
$s \sim (\sim x_1 * x_2 + x_1 * \sim x_2) * x_3 + (\sim x_1 * x_2 + x_1 * \sim x_2) * \sim x_3$	$x[1,1] = \sim x_1$	
$s \sim (x[1,1] * x_2 + x_1 * \sim x_2) * x_3 + (\sim x_1 * x_2 + x_1 * \sim x_2) * \sim x_3$	$x[2,1] = x[1,1] * x_2$	
$s \sim (x[2,1] + x_1 * \sim x_2) * x_3 + (\sim x_1 * x_2 + x_1 * \sim x_2) * \sim x_3$	$x[1,2] = \sim x_2$	
$s \sim (x[2,1] + x_1 * x[1,2]) * x_3 + (\sim x_1 * x_2 + x_1 * \sim x_2) * \sim x_3$	$x[2,2] = x_1 * x[1,2]$	
$s \sim (x[2,1] + x[2,2]) * x_3 + (\sim x_1 * x_2 + x_1 * \sim x_2) * \sim x_3$	$x[3,1] = x[2,1] + x[2,2]$	
$s \sim x[3,1] * x_3 + (\sim x_1 * x_2 + x_1 * \sim x_2) * \sim x_3$	$x[4,1] = \sim x[3,1]$	
$s = x[4,1] * x_3 + (\sim x_1 * x_2 + x_1 * \sim x_2) * \sim x_3$	$x[5,1] = x[4,1] * x_3$	
$s = x[5,1] + (\sim x_1 * x_2 + x_1 * \sim x_2) * \sim x_3$	$x[1,3] = \sim x_1$	$x[1,1]$
$s = x[5,1] + (x[1,1] * x_2 + x_1 * \sim x_2) * \sim x_3$	$x[2,3] = x[1,1] * x_2$	$x[2,1]$
$s = x[5,1] + (x[2,1] + x_1 * \sim x_2) * \sim x_3$	$x[1,3] = \sim x_2$	$x[1,2]$
$s = x[5,1] + (x[2,1] + x_1 * \sim x_2) * \sim x_3$	$x[2,3] = x_1 * x[1,2]$	$x[2,2]$
$s = x[5,1] + (x[2,1] + x[2,2]) * \sim x_3$	$x[3,2] = x[2,1] + x[2,2]$	$x[3,1]$
$s = x[5,1] + x[3,1] * \sim x_3$	$x[1,3] = \sim x_3$	
$s = x[5,1] + x[3,1] * x[1,3]$	$x[4,2] = x[3,1] * x[1,3]$	
$s = x[5,1] + x[4,2]$	$x[6,1] = x[5,1] + x[4,2]$	
$s = x[6,1]$		
$c = (\sim x_1 * x_2 + x_1 * \sim x_2) * x_3 + x_1 * x_2$	$x[1,4] = \sim x_1$	$x[1,1]$
$c = (x[1,1] * x_2 + x_1 * \sim x_2) * x_3 + x_1 * x_2$	$x[2,3] = x[1,1] * x_2$	$x[2,1]$
$c = (x[2,1] + x_1 * \sim x_2) * x_3 + x_1 * x_2$	$x[1,4] = \sim x_2$	$x[1,2]$
$c = (x[2,1] + x_1 * x[1,2]) * x_3 + x_1 * x_2$	$x[2,3] = x_1 * x[1,2]$	$x[2,2]$
$c = (x[2,1] + x[2,2]) * x_3 + x_1 * x_2$	$x[3,2] = x[2,1] + x[2,2]$	$x[3,1]$
$c = x[3,1] * x_3 + x_1 * x_2$	$x[4,3] = x[3,1] * x_3$	
$c = x[4,3] + x_1 * x_2$	$x[1,4] = x_1 * x_2$	
$c = x[4,3] + x[1,4]$	$x[5,2] = x[4,3] + x[1,4]$	
$c = x[5,2]$		

3.3 Composition of Logical System from Substrings

Composition of a logical system comes out of a system of substrings obtained by a decomposition of an algebraic expression (section 3.2). The result of the composition is a system of algebraic expressions of all system modules output functions. Except the system of substrings, the degree of modules, which the composed circuit should consist of, is the input for the algorithm. Determination of module degrees depends upon the task which is the de/composition done for. The module levels can be equal

or different for composed modules. The modules composition must be without overlapping of modules.

An algorithm of a logical module composition from substrings representing one-level circuits can start from primary outputs, primary inputs of circuit or points which present inputs for more than one modules. One algorithm of a logical module composition, which starts from primary outputs, is following (example of an application of the presented algorithm for a full adder circuit is presented in the Table 2):

Table 2. Composition of full adder algebraic expression from substrings

Function	Substitution
$s = x[6,1]$ $s = x[5,1] + x[4,2]$ $s = x[4,1]*x3 + x[4,2]$ $s = \sim x[3,1]*x3 + x[4,2]$ $s = \sim x[3,1]*x3 + x[3,1]*x[1,3]$ $s = \sim x[3,1]*x3 + x[3,1]*\sim x3$	$x[6,1] = x[5,1] + x[4,2]$ $x[5,1] = x[4,1]*x3$ $x[4,1] = \sim x[3,1]$ $x[4,2] = x[3,1]*x[1,3]$ $x[1,3] = \sim x3$
$x[3,1] = x[2,1] + x[2,2]$ $x[3,1] = x[1,1]*x2 + x[2,2]$ $x[3,1] = \sim x1*x2 + x[2,2]$ $x[3,1] = \sim x1*x2 + x1*x[1,2]$ $x[3,1] = \sim x1*x2 + x1*\sim x2$	$x[2,1] = x[1,1]*x2$ $x[1,1] = \sim x1$ $x[2,2] = x1*x[1,2]$ $x[1,2] = \sim x2$
$c = x[5,2]$ $c = x[4,3] + x[1,4]$ $c = x[3,1]*x3 + x[1,4]$ $c = x[3,1]*x3 + x1*x2$	$x[5,2] = x[4,3] + x[1,4]$ $x[4,3] = x[3,1]*x3$ $x[1,4] = x1*x2$

INPUT:

A system of algebraic expression substrings that contain exactly one logical operation (section 3.2).

Degree of modules, which the resulting circuit should consist of.

OUTPUT:

System of algebraic expressions that represent circuit output functions and system modules output functions.

BODY:

9. Let n be the depth of function embedding.

Let m be the depth of modules function embedding (degree of modules) that resulting circuit should consist of.

If $(n \bmod m) = 0$, let $m_temp = n - m$, $k = m$, else $m_temp = n - (n \bmod m)$, $k = (n \bmod m)$.

10. Beginning from a variable of the highest level, continuously substitute all variables of higher level than m_temp by strings that contain variables of lower level.

11. In this expression, replace variables with level lower than m_temp by substitution of particular substrings. This substitution should be performed until the number of substitutions in substring in which the corresponding variable belongs to is not higher than k . The obtained expression presents the resulting function of module.

12. In this expression, if exists no variable $x[p,q]$, $p > 0$ then stop else:

- 4.1. For every variable $x[p,q]$, $p > 0$ do the following:
 - if $p \geq m$, let $m_temp = p - m$, else $m_temp = 0$.
- 4.2. Let $k = m$, proceed to step 2.

Functions of outputs $s = x[6,1]$, $c = x[5,2]$ gained by substitutions presented in section 3.2 are inputs for the algorithm. We chose $m = 3$, because the circuit has to be realized by logical elements NOT, AND, OR, and gain a circuit consisting of three-level modules this way.

We can find out that the module M1 with output $x[3,1]$ is isomorphic to the module M2 with output s , what becomes interesting from the diagnostic point of view [5]. The way of determination of identical and isomorphic modules is described in section 4. Structural scheme of a de/composed full adder circuit is shown in Fig. 1.

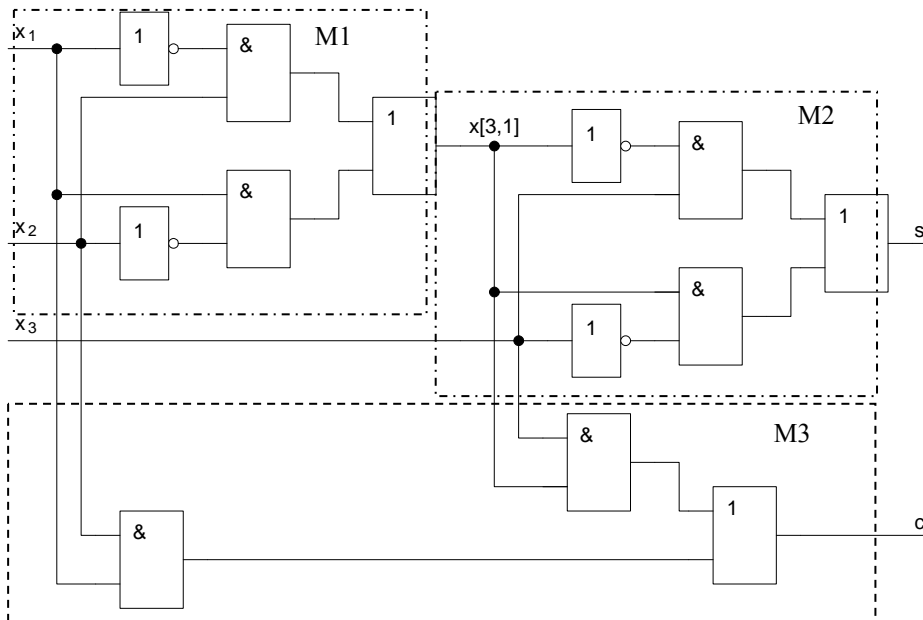


Fig. 22. Structural scheme of a de/composed full adder circuit.

4 Determination of Identical and Isomorphic Circuits

Determination of identical circuits is executed in the process of corresponding strings decomposition to one-level substrings (section 3.2). By comparison of expressions obtained this way, we can tell if the given circuits are identical.

Determination of isomorphic circuits is executed after system modules composition (section 3.3). For the determination of isomorphism of the strings it is necessary to find out, if the conditions of identity of corresponding operation (type of the operator and number of operand) and also the condition of variables substitution

are fulfilled. Detection of the last mentioned condition is very demanding, because the number of different substitutions is equal to the number of variables permutations. That's why the condition of substitution is detected only when all other conditions are fulfilled.

The number of substitutions can be lowered, if we consider only permutations that present only those variables that are inputs of the logical operations with the same operator and same number of variables.

One algorithm for decomposition of circuit into modules and for modules isomorphism determination was implemented in [6], [7].

5 Conclusion

Decompositions of logical systems specified by algebraic expressions of its function are proposed in the contribution. The degree of modules is input of algorithm of modules composition. We can repeat modules composition with different degree of modules and look for suitable decomposition.

Acknowledgments

This work is the result of the project implementation: Development of the Center of Information and Communication Technologies for Knowledge Systems (ITMS project code: 26220120030) supported by the Research & Development Operational Program funded by the ERDF.

References

1. Korečko, Š., Hudák, Š., Šimoňák, S.: Formal Methods Integration for Design and Analysis of Time-critical Systems. *Informatika*, Bratislava, pp. 211--216 (2003)
2. Chladný, V., Havlice, Z., Szaniszló, P.: Modeling Tools Description Language. In *Proceedings of the International scientific conference MOSIS in Rožnov pod Radhoštěm, Czech Republic*, pp. 107--112 (2000)
3. Bača J., Chladný V., Giertl J.: Formal Specifications and Decomposition of Logic Systems for Purposes of Analysis, Synthesis and Diagnostics, *Problemy programirovaniya* 16, 2-3, pp. 102--107 (2004)
4. Bača, J.: Decomposition of Logic Circuits. In *Proceedings of International Conference Electronic Computers and Informatics '98*, FEI TU Košice-Herľany, pp.100--103 (1998)
5. Hlavička, J., Kottek, E., Zelený, J.: *Diagnostika elektronických číslicových obvodů*. SNTL, Praha (1982)
6. Hlinka, V.: *Dekompozícia logických obvodov*. Bakalárska práca, Košice, Technická univerzita v Košiciach, Fakulta elektrotechniky a informatiky (2011)
7. Dzubajová, B.: *Dekompozícia logických obvodov*. Bakalárska práca, Košice, Technická univerzita v Košiciach, Fakulta elektrotechniky a informatiky (2011)

Author Index

Alferov, Eugene.....	121	Letychevskiy, Olexander	71
Alferova, Lyudmila	121	Liutenko, Iryna.....	147
Alobaidi, Mizal.....	17	Lomakovska, Ganna	215
Antoniou, Grigoris.....	12	Lvov, Michael.....	188
Bača, Ján.....	344	Lyaletski, Alexander	290
Bădică, Costin.....	247	Manzhula, Anna.....	207
Bădică, Costin.....	248	Mayr, Heinrich C.	4
Batyiev, Andriy	17	Mezghiche, Mohamed.....	275
Besedina, Maryna	96	Mykhailenko, Hlib	33
Borue, Sergey	164	Nagy, Michal	128
Borue, Yuriy	244	Nikitchenko, Mykola	
Cherednichenko, Olga	147 4, 56, 273, 274, 296	
Cochez, Michael.....	249	Omelchenko, Nadiya.....	215
Davidovsky, Maxim	15, 245	Ostrovski, Alexei	176
Davidovsky, Vladimir.....	245	Payentko, Tanya.....	257, 258
Djeddai, Selma.....	275	Peschanenko, Vladimir ...	71, 273, 274
Dobrovolsky, Gennadiy	245	Petrushko, Basyl	82
Doroshenko, Anatoliy.....	112	Protsenko, Galyna	215
Ermolayev, Vadim.....		Radetskiy, Igor.....	176
.....4, 15, 164, 244, 247, 248		Shkilniak, S.S.....	296
Gavrilova, Lyudmila.....	199	Slabospitskaya, Olga.....	155
Gritsyuk, Valentina.....	134	Spivakovsky, Aleksander.....	4, 121
Hudák, Štefan	321	Strecker, Martin	13, 273, 274, 275
Ivanov, Ievgen	312	Tatarintseva, Olga	244
Keberle, Natalya	164, 222	Terziyan, Vagan.....	247, 249
Khizhnyak, Inna.....	228	Terziyan, Vagan.....	248
Khristenko, A.....	267	Todoriko, Olga.....	245
Klimenko, Vitaly	290	Tolok, Vyacheslav	15
Klionov, Dmitriy M.....	102	Tukalo, Sergey	82
Kobets, Vitaliy.....	4, 257, 258, 259	Tymofieiev, Valentyn G.	56
Kolesnyk, Andrii	155	Utkin, Ivan V.	222
Kovalenko, Roman	33	Weissblut, A. J.	267
Kravtsov, Dmytro	44	Yangolenko, Olga	147
Kruglyk, Vladyslav	188	Zadorozhna, Natalya	82
Kryukov, Sergey.....	257, 258	Zaretska, Iryna	33, 96
Kulankhina, Oleksandra	33	Zavileysky, Mikhail	4
Kushnir, Nataliya.....	207	Zaytseva, Tatyana	236
Kutetsky, Dmitry	134	Zhereb, Kostiantyn.....	112
Lavrishcheva, Ekaterina	176	Zholtkevych, Grygoriy.....	4, 17
Letichevsky, Alexander	71		

