Approach to E-Learning Fundamental Aspects of Software Engineering

Ekaterina Lavrischeva[1], Alexei Ostrovski[1], and Igor Radetskiy[1]

[1]Institute 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).

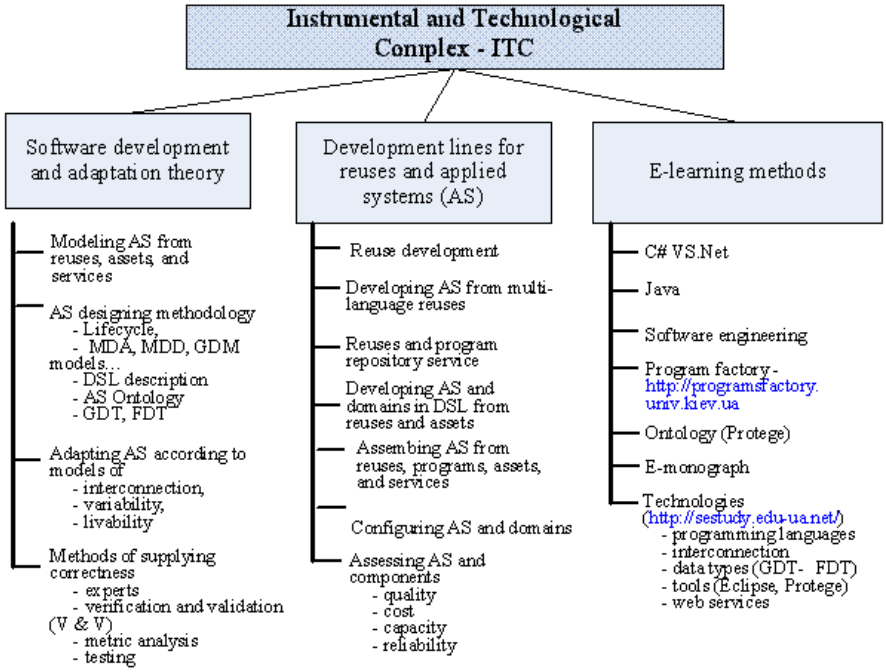

Fig. 6. Structure of the ITC.

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.

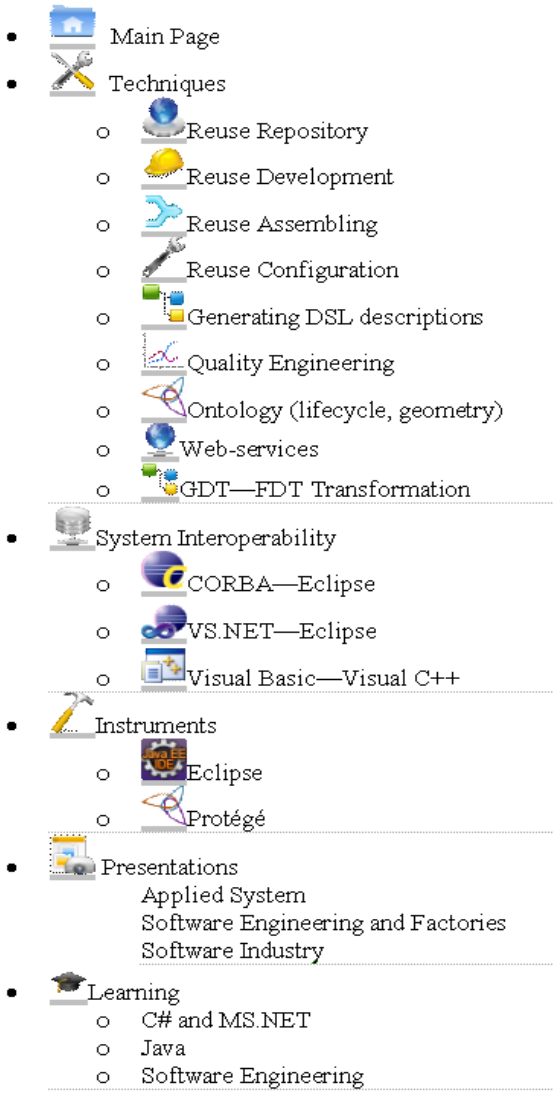

Fig. 7. Keywords in the main page of the web site

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.

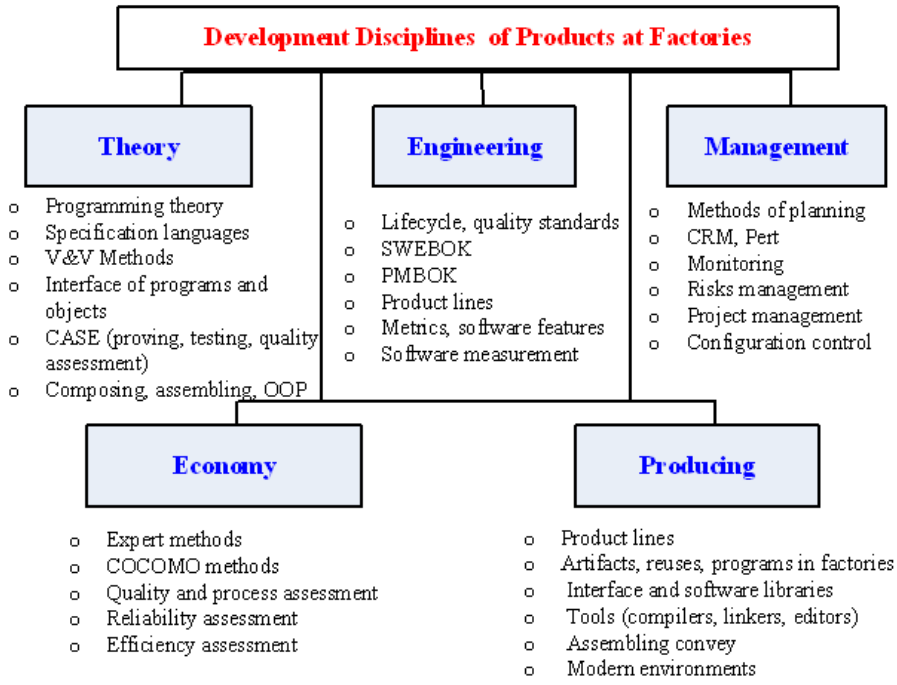

Fig. 8. Classification of software engineering disciplines.

- 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).

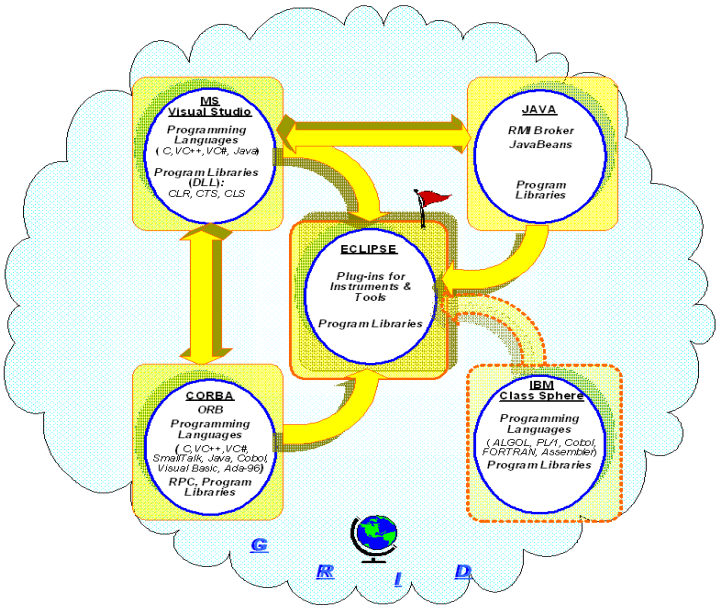

Fig. 9. Structure of interconnection between general environments.

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. Akademperiodika, Kiev (2009) (in Ukrainian)

8. Lavrischeva, K.: Formal Fundamentals of Component Interoperability in Programming. In: Cybernetics and Systems Analysis, vol. 46, no. 4, pp. 639–652. Springer, Heidelberg (2010)
9. Lavrischeva, 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., Lavrischeva, E., Suslov, V.: Foundation of Quality Engineering of Software Systems. Akademperiodika, Kiev, 2nd ed. (2007) (in Russian)
12. Lavrischeva, 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. Lavrischeva, E.: Classification of Software Engineering Disciplines. In: Cybernetics and Systems Analysis, vol. 44, no. 6, pp. 791–796. Springer, Heidelberg (2008)
14. Lavrischeva, 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. Lavrischeva, E.: Software Engineering. Textbook. Akademperiodika, Kiev (2008) (in Ukrainian)
16. Lavrischeva, 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. Lavrischeva, 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. Lavrischeva, 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., Lavrischeva, 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. Lavrischeva, E.: Theoretical and Applied Aspects of Software Systems Development. In: TAAPSD'2010, pp. 274–285. Kiev (2010) (in Ukrainian)
24. Lavrischeva, 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., Lavrischeva, E., Shevchenko, V.: On Scientific Software Industry. Technical report, Conf. Theoretical and Applied Aspects of Cybernetics (2011) (in Ukrainian)