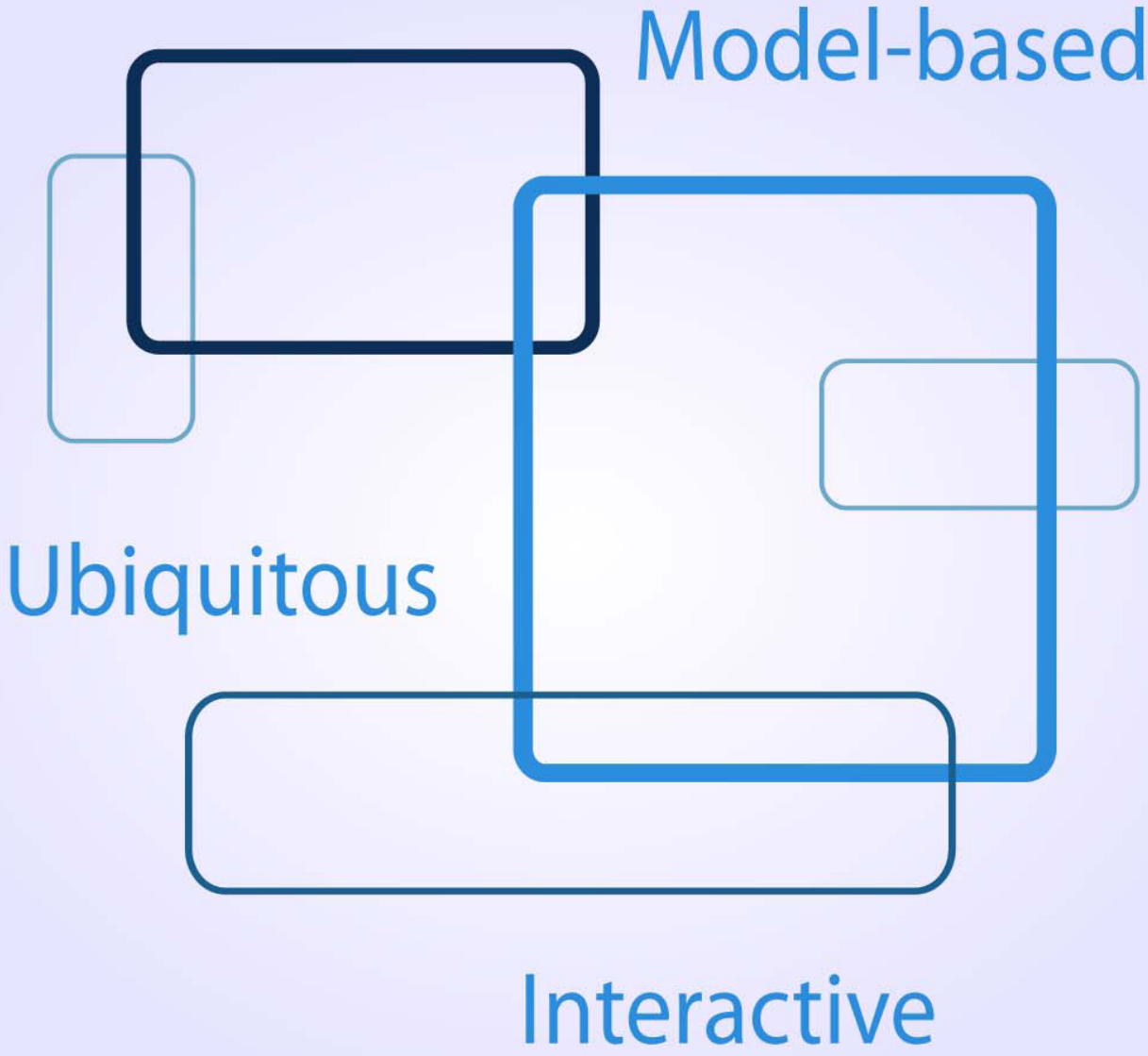

Modiquitous 2012

Proceedings of the 2nd International Workshop on Model-based Interactive Ubiquitous Systems

SES

MMT

TECHNISCHE UNIVERSITÄT DRESDEN

Proceedings of Modiquitous Workshop

Publication Online-CEUR Proceedings (CEUR-WS.org)
CEUR-WS Vol-947
Publication Year 2012

V.i.s.d.P:
Jun.-Prof. Dr. Thomas Schlegel
Junior Professorship for Software Engineering of Ubiquitous Systems
Institute for Multimedia and Software Technology
Technische Universität Dresden
01062 Dresden
Germany

CONTENTS

1 WORKSHOP ORGANIZERS

1.1 Thomas Schlegel

Technische Universitat Dresden

01062 Dresden

Germany

thomas.schlegel@tu-dresden.de

Thomas Schlegel is Junior Professor for Software Engineering of Ubiquitous Systems at the Institute of Software and Multimedia Technology of the Technical University of Dresden. Before he joined the University of Stuttgart as team leader for Interactive Systems, he worked as senior researcher and research project leader at Fraunhofer IAO from 2002, where he served as research cluster leader in the European Network of Excellence I*PROMS and led various national and international research projects. He received his PhD in engineering from the University of Stuttgart. He is author and co-author of 60 scientific publications and serves as reviewer and committee member for diverse international conferences.

1.2 Romina Kühn

Technische Universitat Dresden

01062 Dresden

Germany

romina.kuehn@tu-dresden.de

Romina Kühn is a research associate at the Junior Professorship for Software Engineering of Ubiquitous Systems at the Technical University of Dresden. Her research interests include interaction concepts, interface design and development, and usability aspects. Public ubiquitous systems and especially systems in public transportation are her main application field.

1.3 Stefan Pietschmann

Technische Universitat Dresden

01062 Dresden

Germany

stefan.pietschmann@tu-dresden.de

Stefan Pietschmann is research associate and Ph.D. student at the Institute of Software and Multimedia Technology of the Technical University of Dresden. He has been actively involved in several research projects in the field of collaborative and context-aware web applications. In the project CRUISe he specifically addresses the model-driven development of adaptive interactive applications based on the idea of a universal service composition.

2 PROGRAMME COMMITTEE

- Uwe Aßmann, Technical University of Dresden, Germany
- Jan van den Bergh, Hasselt University, Belgium
- Birgit Bomsdorf, Hochschule Fulda, Germany
- Raimund Dachselt, Otto von Guericke University of Magdeburg, Germany
- Florian Daniel, University of Trento, Italy
- Alfonso Garcia-Frey, University of Grenoble, France
- Geert-Jan Houben, Technical University of Delft, Netherlands
- Heinrich Hussmann, Ludwig-Maximilian University Munich, Germany
- Sevan Kavaldjian, Vienna University of Technology, Austria Programme Committee
- Gerrit Meixner, DFKI,Germany
- Philippe Palanque, University of Toulouse, France
- Fabiò Paterno, CNR-ISTI, Italy
- Michael Raschke, University of Stuttgart, Germany
- Dirk Roscher, Technical University Berlin, Germany
- Enrico Rukzio, University Duisburg-Essen, Germany
- Stefan Sauer, University of Paderborn, Germany
- Thomas Springer, Technical University of Dresden, Germany
- Gerhard Weber, Technical University of Dresden, Germany
- Anette Weisbecker Fraunhofer IAO, Stuttgart, Germany
- Jürgen Ziegler, University Duisburg-Essen, Germany

3 INTRODUCTION

Ubiquitous systems today are introducing a new quality of interaction both into our lives and into software engineering. Systems become increasingly dynamic making frequent changes to system structures, distribution, and behavior necessary. Also, adaptation to new user needs and contexts as well as new modalities and communication channels make these systems differ strongly from what has been standard in the last decades.

Models and model-based interaction at runtime and design-time form a promising approach for coping with the dynamics and uncertainties inherent to interactive ubiquitous systems (IUS). Hence, this workshop discussed how model-based approaches can be used to cope with these challenges. Therefore, it covers the range from design-time to runtime models and from interaction to software engineering, addressing issues of interaction with and engineering of interactive ubiquitous systems.

The MODIQUITOUS workshop was intended to discuss challenges and possible solutions of the EICS community to design and runtime aspects of interactive ubiquitous systems with a focus on model-based approaches. It aims to bring together researchers and practitioners focused on different problems of IUS.

4 THEME, GOALS, AND RELEVANCE

Model-based interactive ubiquitous systems form a new promising yet challenging domain within the scope of the Engineering of Interactive Computing Systems (EICS) conference. This workshop is intended to discuss these challenges and possible solutions of the EICS community to design and runtime aspects of interactive ubiquitous systems with a focus on model-based approaches. The related problem space becomes clear when looking at typical future scenarios: users will not only carry their data but also their applications and profiles with them. This may mean switching from planning a project on a desktop system to a collaborative setting in a meeting and further to a mobile or public display setting where a mobile device is used for creating sketches for the first steps in the project. Consequently, applications will evolve from device-oriented to emergent cyber-physical and ubiquitous software in a broad sense, forming interactive and socio-technical systems. This opens manifold possibilities, but also a number of research problems regarding both the development process and the execution environment for those kinds of applications.

The MODIQUITOUS workshop is intended to provide a basis for discussion the adequate solution space. Therefore, it aims to bring together researchers and practitioners focused on different challenges of IUS, including:

- Model-driven architecture (MDA) in the context of IUS
- Advantages and potential problems of using MDA in the IUS domain
- Domain and Meta models for IUS, specifically for IUS-related aspects like interaction, different modalities, dynamic distribution, context?awareness, etc.
- Domain-specific models for IUS
- Model-driven generation of (intelligent) IUS
- Model-to-model and model-to-code transformations for IUS development
- Model-driven development and execution architectures, i.e., runtime systems for IUS
- Tools and frameworks for supporting the model-driven development of IUS
- Concepts for context-awareness and self-adaptation of IUS at the model and runtime level
- Software and Usability Engineering aspects in the context of model-based IUS
- Innovative ideas and novel application solutions for new interactive ubiquitous settings, e.g., from the fields of mobile computing, pervasive computing and social software
- Studies on interaction concepts on IUS

- Requirements, insights and experiences from existing mobile and pervasive settings

All these topics are of high relevance to a big part of the EICS community as their use is not restricted to ubiquitous systems and will show new ways for many kinds of new systems like mobile device settings, pervasive computing and social software.

5 PROGRAM

2nd Workshop on Model-Based Interactive Ubiquitous Systems
Copenhagen, Denmark - June 25, 9:00-17:00

8:30 ***Arrival and Registration***

9:30 ***Welcome and Introductions***
Introductory statements by the organizers and brief introduction by each participant

9:15 ***Discussion and Topic Definition***
Discussion and definition of hot research topics

10:30 ***Coffee Break***

11:00 ***Paper Presentations***

- Peter Forbrig and Michael Zaki
  Models and Patterns for Smart Environments
- Steven Houben, Morten Esbensen and Jakob Bardram
  A Situated Model and Architecture for Distributed Activity-Based Computing
- Uwe Laufs, Christopher Ruff and Jan Zibuschka
  Model-based support for energy efficient production in SME

12:30 ***Lunch***

14:00 ***Paper Presentations***

- Martin Franke, Diana Brozio and Thomas Schlegel
  Towards a flexible control center for cyber-physical systems
- Maximilian Kintz
  A Semantic Dashboard Description Language for a Process-oriented Dashboard Design Methodology

- Georg Püschel, Ronny Seiger and Thomas Schlegel

  Test Modeling for Context-aware Ubiquitous Applications with Feature Petri Nets

15:30 ***Coffee Break***

16:00 ***Group Work***
Discussion of the selected topic, e.g., identification of research roadmap items

16:30 ***Discussion***
Plenum discussion and topic integration

17:00 ***End***

6 ACCEPTED PAPERS

The following papers were accepted:

- Models and Patterns for Smart Environments
- A Situated Model and Architecture for Distributed Activity-Based Computing
- Model-based support for energy-efficent poduction in SME
- Towards a flexible control center for cyber-physical systems
- A Semantic Dashboard Description Language for a Process-oriented Dashboard Design Methodology
- Test Modeling for Context-aware Ubiquitous Applications with Feature Petri Nets

Models and Patterns for Smart Environments

Peter Forbrig
University of Rostock, Department of Computer Science
Albert-Einstein-Str. 22
D-18051 Rostock, Germany
+49 381 498 7620
peter.forbrig@uni-rostock.de

Maik Wurdel, Michael Zaki
University of Rostock, Department of Computer Science
Albert-Einstein-Str. 22
D-18051 Rostock, Germany
+49 381 498 7431
michael.zaki@uni-rostock.de

ABSTRACT
In a given smart meeting room, several users are supposed to cooperate together while employing static and dynamic heterogeneous devices. The goal of such environments is to deliver proper assistance to the users while performing their tasks. Thus, task models are an appropriate starting point for those environments. Those models give the developers the opportunity to focus on the users and their tasks. Tasks are not independent from available tool, locations and acting persons. Therefore, other models have to be developed and linked to the task model in order to truly illustrate how the tasks are executed in those environments. The paper discusses the application of the language CTML that was designed for this purpose. Furthermore, the usage of patterns for supporting the development of models for smart environments will be discussed.

Keywords
Smart environment, HCI, task model, context, goal pattern, task pattern

INTRODUCTION
The idea of ubiquitous computing goes back to Marc Weiser. According to Weiser's vision [11], devices are weaving themselves into everyday life, allowing people to fully concentrate on performing their tasks, while hiding their existence and complexity. A smart environment (SE) tries to analyze user behavior and tries to provide appropriate assistance. Within the context of a meeting scenario the presenter should concentrate on the talk, while the SE is responsible for offering convenient assistance by adjusting the projector, loading the necessary files and capturing audiovisual data for meeting documentation if needed. In the best case no direct interaction is necessary. Implicit interaction like going to the presentation area is enough to present the slides of the speaker. Experiences show that [5] that the quality of support can be increased if some information is given to the system. Most important are the tasks the users want to perform within the environment.

Task models are an appropriate starting point for interactive processes development. The application of task models for smart environments is discussed in [4] We will shortly introduce the concept of task models and an own collaborative task-modeling language will be presented. Afterwards we introduce our ideas of using patterns.

MODELLING COOPERATION WITH TASK TREES
Currently CTT [7] is one of the most referred notations for task models. Tasks are arranged hierarchically, where more complex tasks are decomposed into simpler sub-tasks. CTT distinguishes between several task types, which are represented by the icon representing the task node. There are abstract tasks, which are further decomposable into combinations of the other task types including interaction, application and user tasks (see Fig. 1 for an overview of the available task types). The task type denotes the responsibility of execution (human, machine, interaction, cooperation with human).

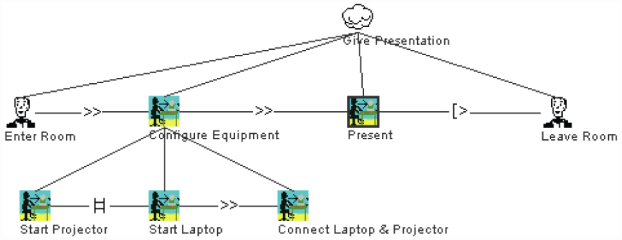

Figure 1 Task model for giving a presentation

Sibling tasks are connected by binary temporal operators. However, unary operators exist that are related to one task only. A complete listing and explanation of the CTT operators can be found in [7]. Operators have precedence orders. These orders are important for interpreting different operators in the same level. The priority of the interleaving (|=|) operator is higher than the enabling operator (>>). The iteration is an example of a unary operator. An example of a CTT model within the context of a smart environment for giving a presentation is shown in Figure 1. It provides the task of giving a presentation. The abstract root task "Give Presentation" is decomposed into four child tasks. The left tasks on the second level of abstraction are connected with the enabling operator (>>) in order to specify that one task has to be performed before the other one can start (e.g., the interactive task "Configure Equipment" can only be performed after having executed the human task "Enter Room"). "Leave Room" can be performed at any time due to the deactivation operator ([>) resulting in a prematurely

P. Forbrig, M. Wurdel, M. Zaki: Models and Patterns for Smart Environments.
Proc. of 2nd International Workshop on Model-based Interactive Ubiquitous System 2012,
Pisa, Italy, June 13, 2011, http://ceur-ws.org/Vol-947

abortion of the currently running task. Furthermore, the task "Configure Equipment" is decomposed into the subtasks "Start Projector", "Start Laptop" and "Connect Laptop & Projector". The third task can only be executed after the first two tasks were performed.

Because of lack of space the model does not specify the details of the presentation. Models can be of course as detailed as necessary.

COLLABORATVE TASK MODELING LANGUAGE

In conjunction with modeling efforts in a smart environment the collaborative task modeling language (CTML) was developed. Despite that the idea was originated in the context of smart environments, it seems to be applicable in a broader range (e.g. Stakeholder-driven process management can be supported in this way). We will shortly discuss the fundamental assumptions and the most significant features of the language.

The design of CTML is based on four fundamental assumptions:

I. **Role-based Modeling**. In limited and well-defined domains the behavior of an actor can be approximated through her role.

II. **Hierarchical Decomposition and Temporal Ordering**. The behavior of each role can be adequately expressed by an associated collaborative task expression.

III. **Causal Modeling**. The execution of tasks may depend on the current state of the environment (defined as the accumulation of the states of all available objects) and in turn may lead to a state modification.

IV. **Individual and Team Modeling**. The execution of individual user tasks may contribute to a higher level team task

Based on these assumptions a collaborative task model is specified in a two-folded manner:

1. Cooperation Model.

   Specifies the structural and behavior properties of the model.

2. Configuration(s).

   Holds runtime information (like initial state, assignment) and simulation / animation configurations.

A cooperation model is presented in Figure 2. Model entities are represented by elements in the inner circle (post fixed with "-1"). Diagrams outside of the inner circle provide more detailed specifications of the corresponding entities (post fixed with "-2"). A specification of a model consists of specifications of roles (e.g., A-1), devices (e.g., B-1), a location (C-1), a domain (D-1) and a team (E-1).

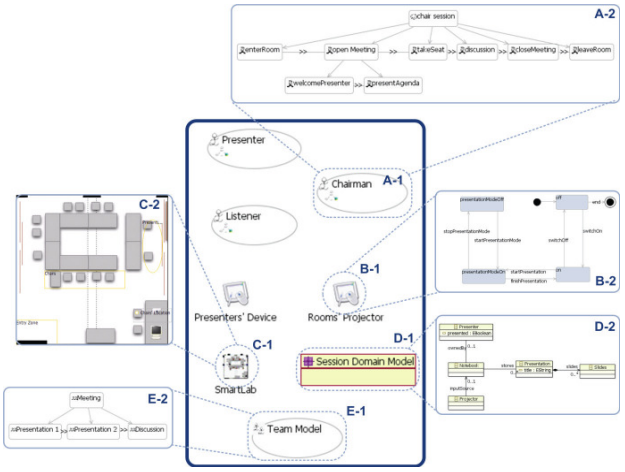

Figure 2 Schematic Cooperation Model for a Meeting

In our following discussion we will focus on roles and their models only. Roles categorize users that have the same capability, responsibility, experience and limitations according to the domain. Thus roles can be considered as abstractions of users sharing the same characteristics. In software engineering roles are often called actors. The potential actions a user is able to perform are determined by his role(s). In CTML a role is associated with a task model (A-2) that is visually represented by a task tree in a CTT-like notation.

CTML allows the dynamic change of roles during runtime, which is not very common in other modeling approaches.

Let us assume that Sheldon acts as Chairman and Leonard acts as Presenter in our smart meeting room. Additionally, there is Penny. She fulfills the roles Presenter and Listener as depicted in Fig. 3.

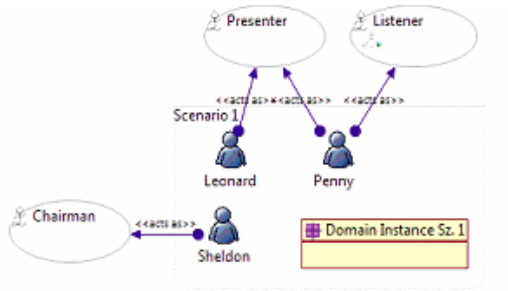

Figure 3 Specific Meeting Configuration

The configuration assigns persons to roles. Additionally, it can be specified that Penny first acts as a presenter and later fulfills the role of a listener. This can be expressed by "Presenter >> Listener" for Penny. A configuration is sometimes also considered as a scenario for which the cooperation model is used.

Sometimes temporal relations are not expressive enough to specify the real constrains between different tasks. This was the reason for introducing textual specifications into

CTML (like OCL [8] for UML). Such specifications allow constrains that involve devices, locations and all other model elements. They are used to specify preconditions and effects of tasks using an OCL-like syntax. For the role 'Chairman' and the role 'Presenter' the preconditions and effects shown in Table 2 make sense.

Table 1 Examples for preconditions and effects

Role	Task	Precondition
1 Presenter	Start presentation	Chairman.oneInstance.AnnounceTalk
2 Chairman	Announce discussion	Presenter.allInstances.EndPresentation
Role	**Task**	**Effect**
1 Presenter	End resentation	self.presented = true
2 Chairman	Announce discussion	Notebook.allInstances.switchOff

A presenter is only allowed to start his presentation after the chairman has announced it (precondition 1). Precondition 2 states that a chairman can only announce a discussion if all presenters have finished their presentations. It might be a little bit difficult to create such kind of specifications but they have the advantage of being readable to some extent. For the expressiveness of such specifications quantifiers are very important. They allow specifying the number of actors or devices (one or all).

After a Presenter ended his talk the corresponding attribute is set to true (effect 1). After the Chairman has opened the general discussion all notebooks in the room are switched off (effect 2).

It is possible to specify activities taking place in a smart environment in a precise way. However, it is sometimes a burden to develop such a specification. The modeling process is complex and time consuming. A promising idea would be to overcome this problem by using existing specifications to build new ones.

Patterns have proved [5] to be a good tool to represent knowledge in software design. They spread through computer science domain despite the fact that they were first discussed in architecture [1]. Additionally, many approaches take benefit of the usage of patterns in the HCI area [21]. Breedvelt-Schouten et al. [3] introduced task patterns that inspired our work. Sinnig [9] provided generic task patterns to be able to adapt a pattern to the context of use.

In a given smart environment numerous actors try to achieve a common goal that can be characterized as team goal. For the meeting room example, the ultimate goal is the efficient exchange of information among the actors in the room. Every task executed by an actor in its role is in a way a contribution to the team goal. It is a step towards this goal. Additionally, the task helps to reach the own individual goal (e.g. to make a good presentation).

A first step to develop patterns in the context of smart meeting rooms was to identify possible team goals (a certain state that the team wants to reach). First results were presented in [25] by providing six abstract team goals. These goals were (I) conference session performed, (II) lecture given, (III) work defended, (IV) topic discussed, (V) debate managed and (VI) video watched.

In the meantime some further patterns were identified. Figure 4 presents one of those patterns in a simplified way. It is a team pattern for discussing phenomena of the climate that was identified by observing meetings in a research institute.

Usually during meetings at this institute there is first a general presentation. Later on participants split into two subgroups and discuss some pictures and data. in two subgroups and at the end the combined results from both groups are presented to the whole plenum.

Unlike the former task patterns approaches it is our goal to integrate the so-called "forcing context" into the pattern specification. The forcing context describes the set of environmental preconditions that have to be fulfilled in order to execute the tasks within the pattern and the set of post-conditions expressing the influence of the execution of those tasks on the state of the environment.

smart environments. Moreover, the main trend nowadays is the design of universally accessible applications.

The pattern description consists of an ID, name, problem, situation, solution, diagram, adaptation variables and referenced patterns. In this paper we will concentrate on the illustrations (in the diagram section of the pattern) that are provided within the pattern example of Figure 5.

The diagram section of the pattern consists of three parts, the task hierarchy, the environmental dependencies and the visualization of the execution constrains.

Currently we are extending the CTML editor which provides an Eclipse-based IDE to build task models by our task pattern application tool. We truly believe that having this pattern library fostered by the tool, the developer will be offered a real assistance while modeling a given scenario in the context of smart environments.

CONCLUSIONS

In this paper we argued for a model-based approach for smart environments. We presented some details of our specification language CTML that allows specifying the tasks of different actors and the cooperation of a team. It is argued to split the specification into a cooperation model and a configuration model. The cooperation model specifies general knowledge of activities of a specific domain. This knowledge is long lasting. The configuration model has to be specified according to the current instance of a session. Who are the participants that take part and which roles do they play?

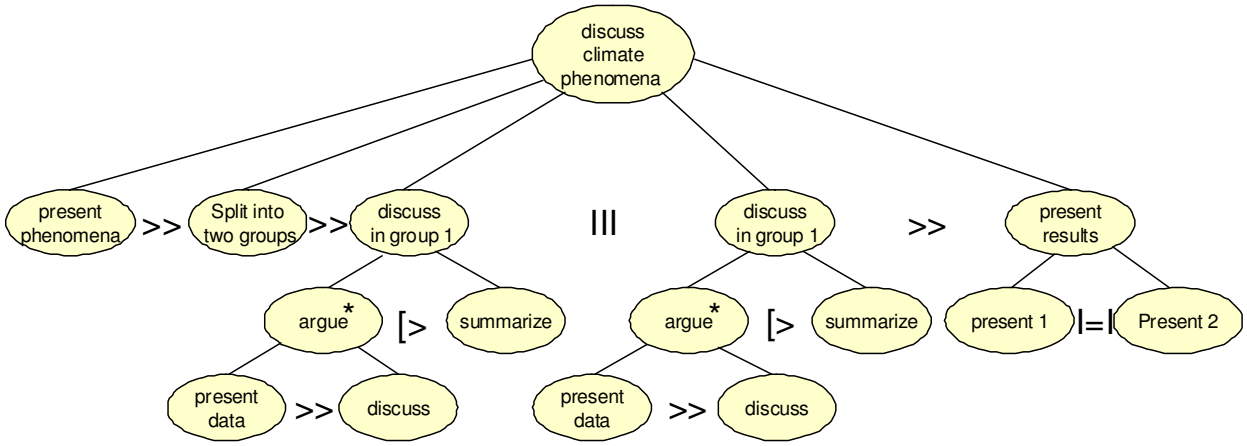

Figure 4 Fraction of a Team Pattern Example

Afterwards, we argued that the need to build the task models, the other environmental and domain models as well as the specification of the relations and constraints between all of the included models dramatically increase the complexity of the modeling process of those environment. Therefore, we suggested the usage of appropriate patterns which aim to provide convenient support to the developer while modeling her scenario.

The animation of our models can support the Bayesian algorithms of a smart environment that try to infer next possible actions of the users based on the sensor data. On the other hand these algorithms can inform the animation of the models that certain preconditions of task are fulfilled.

REFERENCES

1. Alexander, C., Silverstien, M.: A Pattern Language. In: Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid F. King und Shlomo Angel (Eds.), Towns, Buildings, Construction, Oxford University Press, New York 1977, ISBN 0195019199
2. Blumendorf, M.: Multimodal Interaction in Smart Environments A Model-based Runtime System for Ubiquitous User Interfaces. Dissertation, Technische Universität Berlin, 2009.
3. Breedvelt-Schouten, I.M., Paterno, F., Severijns, C.: "Reusable Structures in Task Models", 1997, In Proceedings of DSV-IS: 225-239.
4. Blumendorf, M., Lehmann,G.,Albayrak, S.: Bridging Models and Systems at Runtime To Build Adaptive User Interfaces. Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems. ISBN: 978-1-4503-0083-4 : 9-18.
5. Gamma, E., Helm, R., Johnson, R., Vlissides, J., Design Patterns. Elements of Reusable Object-Oriented Software. Addison Wesley, 1995
6. Giersich, M., Forbrig, P., Fuchs, G., Kirste, T., Reichart, D., and Schumann, H.: Towards an Integrated Approach for Task Modeling and Human Behavior Recognition, Proc. HCII 2007, p. 1109-1118, ISBN: 978-3-540-73105-4.
7. Mori, G., F. Paternò and C. Santoro: CTTE: Support for Developing and Analyzing Task Models for Interactive System Design, IEEE Trans. Software Eng. 28(8), 2002, p. 797-813.
8. OCL: http://www.omg.org/technology/ documents/modeling_spec_catalog.htm#OCL.
9. Sinnig, D.: The Complexity of Patterns and Model-based Development, Computer Science. Montreal, Concordia University. (Thesis 2004).
10. Tidewell, J.: Interaction Design Patterns: Twelve Theses, PLoP'98, Monticello, Illinois, In. Proc. Conference on Pattern Languages of Programming, 1998.
11. Weiser, M.: The Computer for the 21st Century. Scientific American, 265, pp.94-104, 1991.
12. Wurdel, M., Sinnig, D., Forbrig, P.: CTML: Domain and Task Modeling for Collaborative Environments. Journal of Universal Computer Science 14, 2008, p. 3188-3201.
13. Zaki, M., Forbrig, P.: User-oriented Accessibility Patterns for Smart Environments. Springer Volume 6761/2011, 319-327, DOI: 10.1007/978-3-642-21602-2_35.
14. Zaki, M., Forbrig, P.: Towards a Pattern Language for Modeling Interactive Applications in Smart Meeting Rooms.

A Situated Model and Architecture for Distributed Activity-Based Computing

Steven Houben, Morten Esbensen and Jakob E. Bardram
The Pervasive Interaction Technology Lab
IT University of Copenhagen
{shou,mortenq,bardram}@itu.dk

ABSTRACT
In this position paper, we introduce the *situated activity model*, an activity theory informed activity-based computing (ABC) model that unites the description of *activity* and *situated contexts* into one computational model. By introducing a unified activity model, we seek to connect cross domain activity systems ranging from desktop systems to pervasive computing and beyond. In this paper, we describe (i) the situated activity model (SAM), (ii) a conceptual description of a generic cloud-based architecture for the prototyping and development of situated activity systems (SAS) and (iii) the value of the situated activity approach in different application domains.

Author Keywords
Activity, Activity-Based Computing, ABC, Situated Activity Model, Activity Theory, Context

ACM Classification Keywords
H.5.m Information interfaces and presentation (e.g., HCI): Miscellaneous.

INTRODUCTION

Activity-based computing (ABC) is a term, originally proposed by Apple Research [15], to define a new interaction paradigm that supports activities that users perform rather then the tools they use. The purpose of task- or activity-based computing is to aggregate relevant resources and tools into a higher level structure, activity (or task), that represents an intention of work. Since its first introduction, many different approaches to ABC have been proposed to structure parallel work, context or even augmented interaction in the world. These approaches focus on different areas in Human-Computer Interaction (HCI), ranging from the classic desktop interface [1, 6, 14, 16] and more recently ubiquitous computing [2, 5, 12].

Activity theory [7] has been introduced to the HCI community to theorize the interaction between humans and computers using activity as a unit of analysis. Over the years, activity theory has been refined into models that provide a design space to translate concepts of activity theory to interaction design [9, 4]. In parallel, a bottom-up approach has been used to embed the theoretical notion of activity in work practices through empirical studies, resulting in guidelines and principles for frameworks or systems [1, 16] and meta models [13]. However, in her reflection on second wave HCI theories [3], Suzanne Bødker argues that the application of second wave theories, including activity theory, did not complete its achievements as it failed to describe the shared mediation and context between different *situations*.

Despite the myriad of both theoretical and system approaches to ABC, there is thus no generic activity structure, which implies that there is no unified approach to building or *connecting* ubicomp ABC systems. Because of this, comparing, evaluating, understanding and prototyping systems is difficult. Moreover, there is a gap between the model of activity (in the form of resources, services and contacts) and the use context of activity (in the form of tools or settings). There is thus a fragmentation in both activity *descriptions* and activity *systems*. As an approach to these problems, this paper first proposes the situated activity model (SAM), an activity theory informed ABC model that unites the description of *activity* and *situated contexts* into one computational construct. Second, it describes a cloud - based architecture that can be used for the development of interconnected situated activity systems (SAS). Finally, the paper discusses the potential value of the model and architecture for different application domains.

ACTIVITY THEORY

Activity theory (AT) is a psychological framework that describes human activity as a relation between the subject (S) (human or group that acts in the world), object (O) (which is acted upon and motivates the activity) and the community (C) (or social strata in which the activity is engaged) [7]. The motivation of activity is projected and reflected into an outcome, which is the contribution of activity.

The S-O-C relation is mediated by instruments, rules and division of labour. First, *instruments* provide the subject with a way to act in the world. They externalize the act in the world through enactment and are shaped by affordance and resistance. Second, *rules* define how the act of the subject is embedded in the social context. It socializes the act in the environment, culture and world. Third, *division of labour* structures the relation between the social strata and the object of the activity. It links the distribution of work among community to the hierarchical motive towards the object.

S. Houben, M. Esbensen, J. E. Bardram: A Situated Model and Architecture for Distributed Activity-Based Computing.
Proc. of 2nd International Workshop on Model-based Interactive Ubiquitous System 2012,
Copenhagen, Denmark, June 25, 2012, http://ceur-ws.org/Vol-947

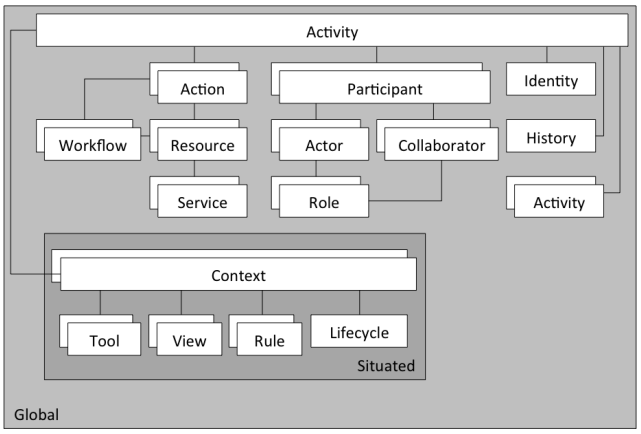

Figure 1. The situated activity model (SAM) is computational representation of activity that describes both global and situated context information.

Activity is not a fixed structure but a dynamic hierarchical interaction between the activity itself driven by motivation, conscious goal-directed actions and unconscious operations that are performed when certain conditions are met. Additionally, Engeström [8] categorized four fundamental processes that are an interwoven into this hierarchy. These processes are: (i) production, (ii) consumption, (iii) exchange and (iv) distribution. In summary, activities are why we do something; actions are what we do it; and operations are how we do [11].

As activity theory moves into its third generation, it has become clear that the unit of analysis is expanding from an individual analysis to a global analysis that comprises not only the individual, community, and artefacts but also the interconnectivity between activity systems. The focus is on networks of interacting activity systems, the dialogues between these systems and the multiple perspectives of these networks of activity [8]. Activity is not an isolated unit of analysis but an integral part of the psychological synthesis of life.

SITUATED ACTIVITY MODEL

The main purpose of Activity-Based Computing (ABC) systems is to lower the amount of configuration work needed to complete a task. We define *configuration work* as the amount of work needed to locate, open and arrange all necessary resources required to complete the objective of an activity. In traditional computing, the user is responsible for the (re-) configuration and maintenance of the workspace to fit the needs of the ongoing activities. ABC systems however can support this *configuration* of activities on three different levels:

1. Interlinking: The first level of configuration is the linking and logical association of activities with actions, resources, actors and community. At this level the resources and requirements of the activity are defined.

2. Situating: The second level situates the deployed activity in the local context through the setting. Several aspects of an activity, defined by the setting of the deployment, are highly coupled with the local context, and this configuration level accounts for this. The situating level describes how the activity can be used in a given context or situation.

3. Visualizing: The third level applies a visualization to the activity. As activities can be consumed on very different devices, different visualization techniques are available. This final level defines how the context of activity is presented.

The situated activity model (SAM) (Figure 1) draws from the basic concepts of AT to provide a computational activity representation that extends the existing activity-based computing [1] model with situated context. It thus merges both the descriptions of the *interlinking* and *situating* configuration level into one unified model that can be used to describe and visualize system-mediated activities on different platforms or systems. The model makes a distinction between *global* and *situated* information.

Global Activity Information

The global activity information is in many ways similar to the original activity-based computing model [1]. It is composed of several interlinked subcomponents that define the content of the activity. Each activity is subdivided into a set of actions, which are subtasks that are part of the activity. Actions structure how actors interact with the different *resources*, such as files and folder, and *services*, such as web services. Additionally, actions can be modelled as *workflows*, which are structured or unstructured sequences of actions that are imposed or defined by the participants.

Participants are human agents that are digitally represented in the system as part of the activity. Participants can be both *actors* and *collaborators*. Actors interact with the system, motivated to complete the *objective* for which the system was designed. They act on the system by using tools that are relevant and related to the activity. Although actors are part of the activity, they own, shape, define, consume, and share activities by interacting with the system. Collaborators are secondary actors that are not directly involved in the *situation* of the interaction of the activity, but contribute to its relevance. They represent external stakeholders that influence and define the object of the activity. Both the capabilities of actors and collaborators are defined through roles. Roles define what actions are accessible and executable by a participant.

Since every activity is an evolving structure, it is embedded into its own history. Each event that occurs within the activity is logged and stored into the activity itself for persistence and reflection. History can be used to track changes in parts the activity, create an awareness on different aspects of the activity or simply to visualize the evolution of the activity. Each activity is uniquely defined by an identity which consists of meta data such as a name, image or description and a unique reference number (e.g. GUID[1]). An activity can be connected to other activities thereby creating a hierarchical relationship or references between activities.

[1]Globally Unique Identifier

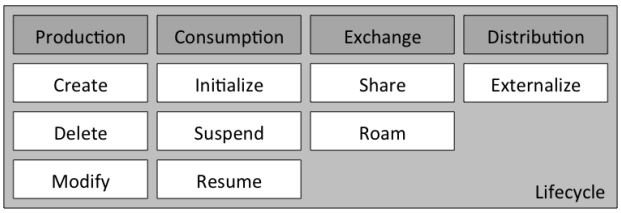

Figure 2. The subprocesses in a SAM lifecycle.

Situated Activity Information

Because a deployment of an activity description is always *situated* and thereby defined by context (environment and social setting in which the activity is used), it is also described by additional context dependent subcomponents. Note that an activity can have multiple contexts depending on different deployments. Tools represent both physical and digital artifacts that allow actors to interact with digital resources accessible through the system. A tool can thus be an application that is shown on a screen or a sensor network that is gathering information about the actor. Besides tools, actors also use physical artifacts that are transformed into tools by augmentation of the system. Tools are the local interface to the actions and digital resources of the activity and thus determine how the activity is consumed.

The setting in which the activity is engaged, is the environment, system or situation in which the activity occurs. It is represented to the user through a view of the setting, providing actors with a level of intelligibility. The view is a mental affordance and interactive tool that describes and demonstrates the capabilities of the system in the setting to the actor. A wall-mounted display or glyph [10] for example, can be a view to represent the capabilities of the setting for the current activity in a pervasive environment while an activity dock or taskbar can be used as a view for desktop systems. The view thus describes how the activity is situated and how it can be consumed. Rules define the policy and access to workflows or actions and are used by the local system to determine how the activity is handled.

Each deployed activity has a lifecycle that determines how the context of the activity is handled by the situated system. The lifecycle consists of the four processes identified by Engeström [7]. The processes are: (i) production (create, delete and modify), (ii) consumption (initialize, suspend and resume), (iii) exchange (share and roam) and (iv) distribution (externalize) (Figure 2).

BUILDING SITUATED ACTIVITY SYSTEMS

By allowing for the deployment of different activity systems that can be interconnected, activities are not confined within one system but can be consumed in all interconnected systems through adaptation of the context. To make the network of interconnected activity systems concrete on a system level, we propose a lightweight but standardized toolkit that can be used to design, prototype and develop interconnected situated activity systems (SAS). The purpose of the toolkit is to provide a lightweight but scalable framework for the development of activity-centric systems. Conceptually, the architecture of the toolkit is composed of two major components: (i) the activity cloud and the (ii) the situated activity system (Figure 3).

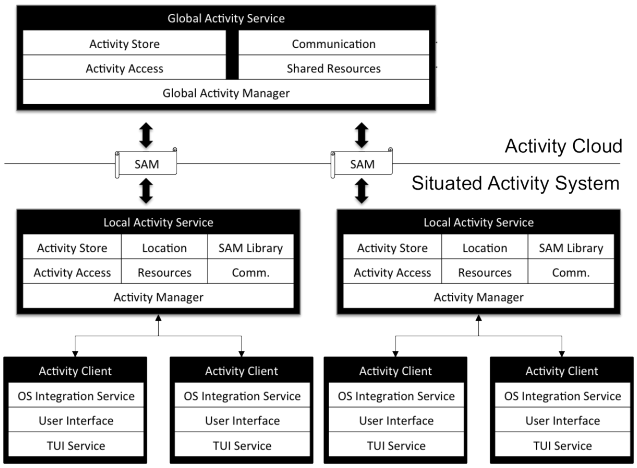

Figure 3. The architecture of the Activity Cloud Toolkit.

Activity Cloud

The main purpose of the activity cloud is to provide a central activity management system that connects different situated activity systems (SAS) to a cloud-based global activity service. The global activity service is composed of different cloud services that are accessible through the global activity manager. Local activity systems can access the cloud manager through a REST-based publish subscribe mechanism. The activity store is used to store SAM descriptions of different SAS. These descriptions are accessible by all connected situated activity systems based on the rules defined in the activity access service. These rules grant local activity systems with authorization to access, modify or manage stored activities. The activity access service also defines how activities and their resources are synchronized with the local activity service. Based on the specification of the local SAS, the cloud synchronization methods can be adjusted. The activity cloud also provides a mechanism to store resources that are shared or exchanged between different SAS's as it allows for the storage of data through a Binary Large Object (Blob) storage. The access to these shared files is managed by the activity manager and should be defined in the local SAS's

Situated Activity System

A situated activity system (SAS) is an SAM-based interaction design system that is composed of two main parts: a local distributed activity service and activity clients. Although both components are architecturally separated, they can be physically used on one device. The local activity service provides support for local activity roaming, sharing and access over all devices that are part of the situated activity system.

The distributed activity manager has a local activity store and access mechanism that has two purposes. First, it is used to manage the activities that are used and shared by different local activity clients. All activity clients are thus connected to

a central repository that handles synchronization, distribution and location tracking of the clients. Second, the local activity service is connected to the global activity service for persistence and real time updates from outside the activity system. Additionally, the local service also provides support for real-time communication between different clients. All devices that are part of the activity system are connected to the local activity manager through an activity client. This client itself is composed of: (i) activity integration services for specific platforms or devices that merge the activity representation into the existing experience, (ii) an activity or task-based user interface (ABC/UI) design and (iii) a tangible or wearable computing layer that connects physical objects with the activity client based on the SAM.

The three components, Global Activity Service, Local Activity Service, and Activity Client correspond in their roles to the three configurations levels as described earlier. The Global Activity Service provides global accessible storage and access to SAM descriptions of different SAS. The Local Activity Service is able to situate activity models in the context of the deployment - e.g. through location tracking information - as defined in the second configuration level. Finally, the Activity Client provides a UI to visualize the activity as described in the third configuration level.

ONGOING AND FUTURE WORK

We are currently in the process of developing the basic toolkit and underlying activity cloud infrastructure. The goal is to test the toolkit (both the model and architecture) by building ABC support for two very different domains: nomadic work in a hospital and global software development collaboration. We choose these settings as previous and current research in the work practices of these domains, allows us to gain great insight in to how work is structured and conducted. Furthermore, we will be able to test our systems in real use settings for end-user validation.

Global Software Development

Global Software Development (GSD) is a development method where the production of software is carried out in multiple locations. The geographical displacement of software teams in GSD however introduces physical, temporal, and cultural challenges. One of the main methods to overcome these challenges is to make extensive use of groupware technologies that allows for collaboration across distances. We have identified a number of different aspects of GSD for which we see an applicability for SAM:

- With the physical distance between teams, all resources should be shared on the network. We can allow this using the cloud activity manager.
- There are clear situated activity systems in GSD, namely those identified with the different physical locations. These different systems however are highly interlinked and share an outcome. We can model the systems through different local activity systems, all connected to the activity cloud.
- An activity-centric approach to GSD allows for the linking of different design artifacts, e.g. source code, design documents or project plans, in to activities
- All physical items, such as PC's tablets, digital Scrum boards etc. can be connected to the local activity manager allowing all time access to relevant information.

Workflow in Hospital Units

Based on observations in the hospital, we describe the workflow of clinicians in a patient ward as nomadic. Nomadic workflow – as compared to mobile work – essentially means that clinicians, instead of being able to carry a laptop or other devices and sit at desks in different location, roam through different departments of the hospital while doing their work. Hence, their work is heavily influenced by collaborations with other clinicians, they regularly use public and shared devices and their general work pattern is characterized by planning and re-planning. Additionally, next to digital artefacts, clinicians also use a large set of medical tools and objects that play an important role in the care of patient.

- Because the entire workflow in a hospital is focused towards patients and their care, it maps on the activity-centric approach of the SAM. All patient information as well as the tools to provide the care can be united into one meaningful structure.
- The nomadic workflow implies that patient information should be ubiquitous, interconnected with stakeholders and available on different locations at different times. The activity cloud can be used to provide this global availability of information. Additionally, it can also be used to provide new activity-centric functionality such as remote patient access or monitoring.
- On a local level, the general patient information can be deployed as a situated activity system. These systems can be an entire department but also a single room that is hosting multiple patients. Because of the link to the global activity cloud, these systems can interact with each other, thereby simplifying collaboration and providing a level of awareness on the patient beyond the local department.
- Physical objects that are related to the patient, such as blood samples, paper documents or medication, can be connected to the activity of the patient.

CONCLUSION

In this paper, we propose a new model and architecture to prototype and develop activity-centric systems. The situated activity model (SAM) unites the description of *activity* and *situated contexts* into one computational representation and can be used to model system-mediated activity. We discussed the theoretical ground of the model, the conceptual architecture of the toolkit and their potential application in different domains. Future work includes supporting multiple platforms, refining the toolkit and deploying situated activity systems in different domains to validate the approach.

ACKNOWLEDGEMENTS

This work was supported by the EU Marie Curie Network iCareNet under grant number 264738.

REFERENCES

1. J. Bardram, J. Bunde-Pedersen, and M. Soegaard. Support for activity-based computing in a personal computing operating system. In *Proc. of CHI '06*, pages 211–220. ACM.
2. J. E. Bardram, J. Bunde-Pedersen, A. Doryab, and S. Sørensen. Clinical surfaces - activity-based computing for distributed multi-display environments in hospitals. In *Proc. of INTERACT '09*, pages 704–717. Springer-Verlag.
3. S. Bødker. When second wave hci meets third wave challenges. In *Proc. of NordiCHI '06*, pages 1–8. ACM.
4. S. Bødker. *Applying Activity Theory to Video Analysis: How to Make Sense of Video Data in Human-Computer Interaction*, chapter 7, pages 148–174. MIT Press, 1996.
5. S. Consolvo, D. W. McDonald, T. Toscos, M. Y. Chen, and J. e. a. Froehlich. Activity sensing in the wild: a field trial of ubifit garden. In *Proc. of CHI '08*, pages 1797–1806. ACM.
6. A. N. Dragunov, T. G. Dietterich, K. Johnsrude, M. McLaughlin, L. Li, and J. L. Herlocker. Tasktracer: a desktop environment to support multi-tasking knowledge workers. In *Proc. of IUI '05*, pages 75–82. ACM.
7. Y. Engestrøm. *Learning by expanding: an activity-theoretical approach to developmental research*. Orienta-Konsultit Oy., 1987.
8. Y. Engeström. Expansive Learning at Work: toward an activity theoretical reconceptualization. *Journal of Education and Work*, 14:133–156, 2001.
9. V. Kaptelinin, B. A. Nardi, and C. Macaulay. Methods & tools: The activity checklist: a tool for representing the "space" of context. *interactions*, 6(4):27–39, 1999.
10. F. Kawsar, J. Vermeulen, K. Smith, K. Luyten, and G. Kortuem. Exploring the design space for situated glyphs to support dynamic work environments. In *Proc of Pervasive'11*, pages 70–78. Springer-Verlag.
11. J. Lave. *Cognition in Practice*. Cambridge University Press, 1988.
12. Y. Li and J. A. Landay. Activity-based prototyping of ubicomp applications for long-lived, everyday human activities. In *Proc of CHI '08*, pages 1303–1312. ACM.
13. T. P. Moran. Unified activity management: Explicitly representing activity in work-support systems. workshop paper at ecscw. In *Proc. of ECSCW 2005*.
14. M. J. Muller, W. Geyer, B. Brownholtz, E. Wilcox, and D. R. Millen. One-hundred days in an activity-centric collaboration environment based on shared objects. In *Proc. of CHI '04*, pages 375–382. ACM.
15. D. A. Norman. *The invisible computer : why good products can fail, the personal computer is so complex, and information appliances are the solution*. 1. mit press paperback ed edition, 1999.
16. S. Voida and E. D. Mynatt. "it feels better than filing": Everyday work experiences in an activity-based computing system. In *Proc. of CHI '09*, pages 259–268. ACM Press.

MODEL-BASED SUPPORT FOR ENERGY-EFFICENT PODUCTION IN SME

Uwe Laufs
Fraunhofer IAO
Nobelstraße 12
Stuttgart BW 70569 Germany
+49 711 970 2120
uwe.laufs@iao.fhg.de

Christopher Ruff
Fraunhofer IAO
Nobelstraße 12
Stuttgart BW 70569 Germany
+49 711 970 2402
christopher.ruff@iao.fhg.de

Jan Zibuschka
Fraunhofer IAO
Nobelstraße 12
Stuttgart BW 70569 Germany
+49 711 970 2401
jan.zibuschka@iao.fhg.de

ABSTRACT
High and still increasing energy costs have led to an increasing relevance of energy efficiency. This paper describes an assistance system that aims to support SME in order to realize energy savings in production and thereby help to increase competitiveness by reducing energy costs. The system is based on a service-oriented architecture. On the server side, several web services provide generic functionality like data management, sensor data and data analysis. In the system's frontend, stakeholder interaction is realized based on Google's Android platform. The system uses data provided by sensors, production orders and additional metadata describing specific properties of the production systems.

A major challenge regarding the realization of such a system is heterogeneity, including different kinds of sensors, a wide range of components within the production systems, third party systems such as ERP systems and mobile devices as frontend for end users. This challenge is addressed by using a model based approach. In addition to the domain models, a stakeholder model is used to customize the user interface for the different stakeholders and a web based user interface is generated to allow system initialization and system configuration.

Keywords
Energy-efficiency, Production, Information Technology, System Architecture, Decision Support, Model based UI Creation

INTRODUCTION

Sustainable production and energy-efficient production are generally seen as the central new paradigms [1] for production research within the next years. More specifically, energy-efficiency has become a more and more important aspect of sustainability, which was originally coined to describe systems allowing for an agile response to competitive challenges [1].

We present a system that provides support for energy-efficient production in SME during both the planning and execution/deployment stages. The system fuses data from sensors reading e.g. energy meters, using metadata and formalized heuristics as well as planning information provided e.g. by an ERP system. From this data the system infers possible actions to reduce energy consumption and is able to notify the stakeholders during production runs via mobile devices. In addition, integrated underlying sensor information is visualized as a strategic decision support tool. Using a model based approach; we create user interfaces from different models to provide basic functionality for system initialization and configuration even if model changes or extensions occur. For end users, a custom user interface for mobile devices is provided.

REQUIREMENTS

There are several technical requirements that have to be addressed by the system's architecture:

Regarding the heterogeneous environment, which includes sensors, ERP systems and several other internal system components, the system's architecture needs to be able to deal with heterogeneity and interoperability aspects. Because of the wide range of application areas in which the system may be used, it has to be extensible and able to integrate into other external environments with minor efforts. Regarding the usage within the production environment, a mobile solution should be provided for end users.

There are also a number of barriers to the adoption of energy-efficiency projects. The barriers are quite heterogeneous, and vary from sector to sector [2][3]. However, offering intermediation through vertical service providers is a well-known strategy for supporting SME [4]. Furthermore, the system should offer a way to identify immediate benefits [5] that could be reaped by e.g. rescheduling tasks.

ARCHITECTURE OVERVIEW

The system is based on a client-server-architecture. On the data layer, mass data such as frequently captured sensor data is stored in a RDBS while domain specific information

U. Laufs, C. Ruff, J. Zibuschka: MODEL-BASED SUPPORT FOR ENERGY-EFFICENT PODUCTION IN SME.
Proc. of 2nd International Workshop on Model-based Interactive Ubiquitous System 2012,
Copenhagen, Denmark, June 25, 2012, http://ceur-ws.org/Vol-947

is managed using ontologies. We provide two separated frontends, a web based for configuration and initialization of the system and a mobile frontend for end users. While the web based UI can directly access data, the mobile UI indirectly accesses the stored data via web services.

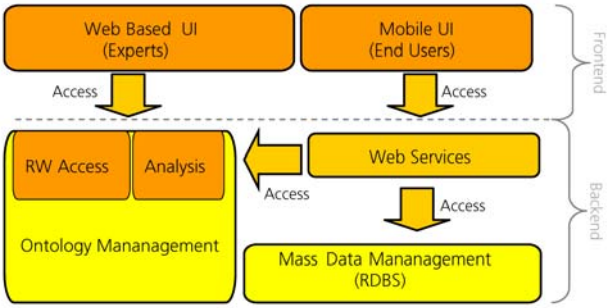

Figure 1: Architecture Overview

BACKEND

Relying on a service-oriented architecture, system's core functionality is realized as a set of key services on the server side. These services are responsible for basic functionalities such as energy meter abstraction, energy consumption monitoring and Analysis, forecasting functionality based on a SARIMA model [6] and Data Management.

Data Management Service

Within the system, different kinds of information have to be collected and managed. Therefore, a data management service is introduced. The service if responsible for providing read/write access to the data and meta-data of the system. Against the background of proprietary implementations and heterogeneous data structures as well as semantic differences in the data provided by energy meters and ERP systems, there is a need for an integrative way to represent this data. We use ontologies to describe the information from various platform-internal and external sources. This approach has already proven to be purposeful, especially in heterogeneous environments [7] [8]. For model realization, we decided to use the web ontology language (OWL) [9]. Based on the Apache Jena Framework [10], ontology individuals can be stored in files and be processed in memory as well as in relational databases. The ontology access and management module is intensively used for storing and retrieving information of the domain description ontologies as well as the user interface customization ontology. The ontology analysis module provides analysis and implements basic analysis functionality for ontology meta-structures such as analysis of ontology concept hierarchies, cardinalities, restrictions and data types. The module also integrates a reasoner [11] which offers a large amount of analysis functionality.

Domain Models

Domain specific Data and meta-data are structured by the following OWL models:

Production systems model: To describe a production system as a whole, a meta-model is provided. It holds a classification of the component types contained in a production system. In addition to types (e.g. motors or lasers), additional properties related to the whole component class (e.g. typical energy consumption ranges, existing component states like on, off, stand-by and valid transitions between states) or relations to other component classes (e.g. interchangeable parts) are included in the model. Based on this information, we are able to infer components of a specific production system that deviate from the typical state in their class.

Production system component model: Furthermore, information about the configuration of the real production system has to be managed. One part of this information is provided by external systems like energy meters or external planning systems. Another part has to be provided during the configuration of the system. The production system component model relies on the high level model described above, which means that a comparison between components that are included in a specific production system and alternative components can be realized.

Manufacturing schedule model: In order to perform scheduling optimizations, we introduce a unified internal scheduling model. The model provides a generic model that contains the relevant scheduling information for the analysis service. In general, any scheduling data that contains the required data (e.g. coming from an ERP system) can be transformed to the internal representation by implementing a translating connector based on a generic interface provided with the platform.

Stakeholder model: Based on the stakeholders of the system, we use a role model which describes which information and which notifications are relevant for which users.

FRONTEND

Regarding the given dynamic environment, user interfaces for basic system interaction have to be adaptable to frequent changes induced by the domain requirements. Therefore, user interfaces for basic system interaction such as the initialisation and configuration of the given production environment in production are created dynamically from the existing domain models and additional configurations for user interface creation.

The system offers a wide range of user interface components, which can be combined to complex web based user interfaces. There are several elements which allow structuring of the user interface (UI) components.

Perspectives: Perspectives are a well known concept, which allow customization of full screen content and also switching between various views.

Containers, Frames and Framesets: Perspectives can be subdivided using Frames and framesets. Within frames, UI elements can be grouped and organized in containers.

Elements: Elements include all non-structuring standalone user interface components such as widgets (buttons, text fields etc).

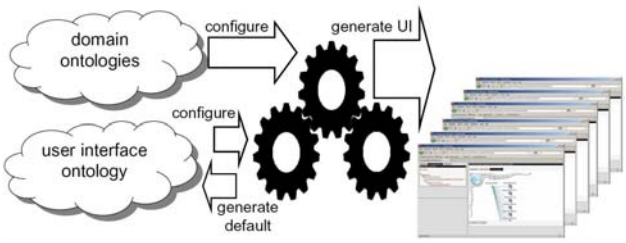

Figure 2: Model Based UI Creation

A User Interface configuration module realizes the functionality required for providing configurable user interfaces based on the domain ontologies. For new or changed domain ontologies, default user interface configurations are created. Missing configurations are created "on the fly" and can be refined afterwards. User interface configurations are realized and stored in user interface ontologies.

The figure below shows a default user interface generated from a non-customized domain ontology. The default user interface ontology provides an instantly working scaffolding mechanism and allows browsing and modification operations on the whole domain ontology.

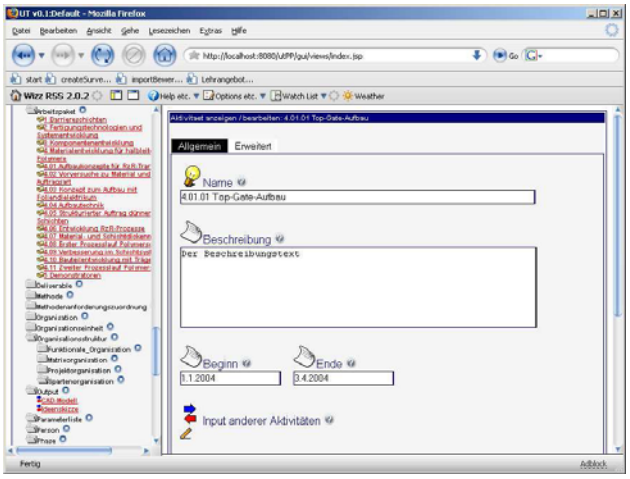

Figure 3: User interface based on default configuration

The generation of default user interface configuration can be adjusted using rules, which are formulated in a UI ontology. Rules can be based on domain ontology content or meta-information, e.g. it can be stated as a rule that specific widgets that visualize textual properties of domain ontologies shall be used based on any of the following criteria:

- Class of the domain ontology concept
- XML datatype of the property
- Defined restrictions defined in the domain ontology
- Defined cardinalities
- Specific URIs of specific domain ontology elements

Specific changes to all parts of the generated user interface configurations can be applied in detail. The figure below shows a configured web based user interface. In the illustrated case, an individual a specific concept is displayed for editing and browsing of related items. The generated configuration is modified regarding the navigation on the left. In addition, a flash based viewer is added which shows the dependencies between the individual and related information.

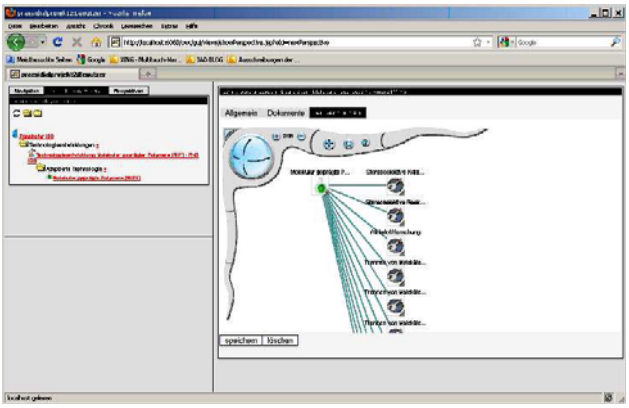

Figure 4: Customized Web User Interface

Mobile Android User Interface

The main objectives of the mobile frontend are to show the current status of energy efficiency of the assembly systems, to notify the stakeholders on identification of any potential to increase energy efficiency and give recommendation on how to achieve savings. To ensure a quick response time to the measures proposed by the assistant system at all times and independent of the current user location, a mobile solution is preferable. This allows the system to be utilized directly in the production environment where the proposed measures can be implemented directly. This also enables the responsible staff to immediately report back the changes made to the production system to the system's back-end, resulting in more up to date, accurate and reliable data. A prototype application for the mobile assistant has been implemented and is being refined using continuous integration and frequent testing using model scenarios.

For the realization of the mobile front-end, the software Framework MT4j is used [12]. MT4j is a java-based open source framework aimed at the creation of visually rich user interfaces which can be interacted with using novel input methods and devices, having a special focus on multi-touch support. First developed for the desktop, it has since been ported to the Android platform, expanding its use onto the various mobile devices relying on that operating system.

The stakeholders of the system have been divided into three different roles: planner and machine operator. The application provides different functionality and options for

the different stakeholders and their respective roles. Roles define the entry point of the application regarding the visibility of production facilities stored in the back-end. Deciders have the ability to navigate through all production sites, assembly machines and their components in detail while the machine operators will typically only see the assembly machines and its components at their location. The roles of the decider or planer are offered an additional configuration menu where cost effectiveness of proposed measures can be calculated depending on certain, configurable variables such as aspired amortization dates or electricity costs. The machine operator is offered an up-to-date list of orders for production. Notifications are tailored to the corresponding user role and can only be read if permitted.

The different stakeholders have to be notified about the potential for energy savings in an appropriate manner. For this purpose the frontend regularly queries the backend whether new notifications are available and then transfers them to the front-end. Notifications include information about the measures to be taken in order to capitalize on the saving potential. Notifications conform to the role concept in such a way that every notification can only be read by a specific role or group of roles. The notifications can generally be divided into two different types: Strategic recommendations are mostly aimed at the role of deciders and planners and contain strategic information, e.g. the availability of alternative assembly components consuming less energy, reducing costs in the long term. Operative recommendations aim at the role of machine operators, these notifications are usually more time critical than strategic recommendations containing information about assembly machines that can be shut down temporarily resulting in immediate energy efficiency gains, for example.

CONCLUSION

We described an IT System that aims to support SME in order to realize energy savings in production and to thereby increase SME competitiveness by reducing energy costs. The system is designed to provide decision support both in the planning and execution phases of production. This functionality relies on the combination of data provided by sensors, production orders and additional metadata describing the properties of the production systems.

A service-oriented architecture is used to allow portability of the system across different manufacturing environments. On the server side, a bundle of key services provides generic functionality like data management, sensor data fusion and state data analysis. Based on models and semantic web technologies, user interfaces create user interfaces for system initialisation and configuration are created dynamically to allow for model changes and extensions with minor efforts.

For end users, mobile clients are provided.

ACKNOWLEDGMENTS

We thank our colleagues from the AssiEff project for their fruitful collaboration, specifically Sebastian Schlund, Stefan Gerlach and Wolfgang Schweizer. This work was financed by Baden-Württemberg Stiftung under Grant AssiEff, „Assistenzsysteme für die auftragsbezogene, energieeffiziente Produktion“.

REFERENCES

1. Jovane F., Koren Y., and Boër C. R., 2003, “Present and Future of Flexible Automation: Towards New Paradigms,” CIRP Annals - Manufacturing Technology, 52(2), pp. 543-560.
2. Grimes P., and Kentor J., 2003, “Exporting the Greenhouse: Foreign Capital Penetration and CO2 Emissions 1980–1996,” journal of world-systems research, 9(2), p. 261–275.
3. Schleich J., 2009, “Barriers to energy efficiency: A comparison across the German commercial and services sector,” Ecological Economics, 68(7), pp. 2150-2159.
4. Albino V., and Kühtz S., 2004, “Enterprise input-output model for local sustainable development--the case of a tiles manufacturer in Italy,” Resources, Conservation and Recycling, 41(3), pp. 165-176.
5. Lockett N. J., and Brown D. H., 2005, “An SME Perspective of Vertical Application Service Providers,” International Journal of Enterprise Information Systems, 1(2), p. 37–55.
6. Olsson M., and Soder L., 2008, “Modeling Real-Time Balancing Power Market Prices Using Combined SARIMA and Markov Processes,” Power Systems, IEEE Transactions on, 23(2), pp. 443-450.
7. Bullinger H.-J., 2006, Fokus Innovation, Carl Hanser Verlag, München.
8. Bügel U., and Laufs U., 2009, “Einsatz innovativer Informations- und Kommunikationstechnologien,” Fokus Technologie. Chancen erkennen, Leistungen entwickeln, Hanser, München.
9. McGuinness D., and van Harmelen F., “OWL Web Ontology Language Overview.”
10. Apache Jena Website, http://incubator.apache.org/jena, visited: April 23th, 2012
11. Clark & Parsia, Pellet Reasoner Homepage, http://clarkparsia.com/pellet, visited: April 23th, 2012
12. MT4j Website, http://www.mt4j.org, visited: April 23th, 2012

Towards a flexible control center for cyber-physical systems

Martin Franke
Technische Universität Dresden
Nöthnitzer Str. 46, D-01187 Dresden
martin.franke@tu-dresden.de

Diana Brozio
Technische Universität Dresden
Nöthnitzer Str. 46, D-01187 Dresden
diana.brozio@tu-dresden.de

Thomas Schlegel
Technische Universität Dresden
Nöthnitzer Str. 46, D-01187 Dresden
thomas.schlegel@tu-dresden.de

ABSTRACT

Today a variety of functionality, like home automation, entertainment or health advice, is running on widely spread consumer hardware, like home servers, mobile phones or consoles. An intelligent interconnection of such hardware forms cyber-physical systems (CPS). This kind of novel systems composes complex ubiquitous systems, in order to connect the physical reality with the digital world. In this paper we describe a possibility to integrate and control sensors and actuators in a seamless manner to an existing system. To achieve the objective, we use semantic model technologies, like ontologies and reasoning over these for this flexible and knowledge-oriented integration of cyber-physical systems. The knowledge is acquired on the one hand by model instances and on the other hand by runtime information and user interactions with the participating devices.

Author Keywords

cyber-physical system; smart space; semantic models; ubiquitous systems

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation (e.g. HCI): Miscellaneous

General Terms

Theory; Design; Performance; Human Factors; Verification.

INTRODUCTION

The development of cyber-physical systems is a complex process. It handles the vision of the "disappearing technologies" and the ubiquitous computing [12] with the help of wireless technologies [4].

Jaroucheh et. al [5] presents requirements for middleware-based context-aware applications that are very similar to cyber-physical systems. These conditions are: coordination of all resources, interoperability and heterogeneity of devices and systems, mobility of the user, autonomous behaviour of the system and potentially auto-discovery of services and devices. In order to achieve this requirements, descriptive models and a middleware, which can handle multiple devices and resources decentralized, are necessary.

The PERSONA project [1] builds one solution for middleware-based context-aware applications. The particularity is the self-organizing infrastructure that handles point-to-point communication with all devices based on service discovery and service adaptation. Components have to register with communication buses to the PERSONA middleware for finding each other and collaborating over these buses. Over these buses, it is also possible to interact in a multi-modal manner with the system. Another outcome of this project is the concept of a context reasoner that aggregate and interpret context information to situations.

Another solution for middleware-based context-aware systems is the Network Automated Machine (NAM) [7]. The NAM is a tool for describing a network of nodes, e.g. devices and provided services in a cyber-physical system. A node is described by its set of physical resources and a set of functional modules that are in turn characterized by a set of provided services, consumed services, consumed context events and provided context events. The service model is based on the IOPE approach of OWL-S [6] that defines a set of input and output parameters, and the precondition and effects of the service. The NAMs can interact with each other over rules, the so called policies, that invokes the specific service on the node.

The central issue of the introduced approaches is the lack of flexibility in the use of multiple devices and interaction concepts. The PERSONA middleware defines communication channels in the time of registration, so the components subscribe for one or more fixed context information from special services. The NAM approach can handle later added cyber-physical components, but is also to fixed in the communication channels of nodes. Thereby we focussed on the aspect of flexibility, based on semantic model descriptions for interaction methods and derived data flow in this work.

The reminder is structured as follows. The second section introduces the VICCI project with its vision and goals. The third section shows a possible architecture concept derived from the requirements for cyber-physical systems. In the

M. Franke, D. Brozio, T. Schlegel: Towards a flexible control center for cyber-physical systems.
Proc. of 2nd International Workshop on Model-based Interactive Ubiquitous System 2012,
Copenhagen, Denmark, June 25, 2012, http://ceur-ws.org/Vol-947

fourth section we outline two scenarios according to our concept. Finally, the fifth section concludes our paper and gives insight to our future work.

THE VICCI PROJECT

The principal purpose of VICCI [11] is the dynamic assisting of the user in cyber-physical systems. This assistant leads to help visualizing and controlling of cyber-physical systems, from Individual Smart Spaces (ISS) towards Smart Communities (SC) [9], in an intuitive, efficient manner.

As motivation of the project deemed the rapid development and ubiquity of technological components, like embedded computers or high-level sensors. The control and the combining of this high technology components is a complex and difficult process. Also an efficient reuse of existing infrastructure in this smart spaces has to implemented.

The control center can be understood as an adaptive, ubiquitous dashboard, that can be viewed from all devices, that are connected to it. The user shall be able to interact with the control center leveraged by different devices and interaction concepts. For this a multi-device interaction is necessary, e.g. the user interact with the same actuator on its PC and mobile device. This ability is acquired by encapsulate functionality in so called Apps. Because of the user-centred approach of the VICCI project, only Apps with an user interface are examined, however they are can run in background without the need to close.

We prefer to use semantic technologies like ontologies and reasoning over these, for automated and knowledge-oriented combining of Apps with their used sensors and actuators. The next section show our architecture concept for the ubiquitous visual frontend and the underlying backend.

ARCHITECTURE CONCEPT

Poovendran [8] describes a major difference between CPS and a regular control system or an embedded system in the use of communications, which adds reconfigurability and scalability as well as complexity and potential instability. CPS still has significantly more intelligence in sensors and actuators as well as substantially stricter performance constraints.

The following chapters introduces the potential frontend- and backend-architectures in VICCI.

Ubiquitous Visual Frontend

Comprehensive cyber-physical systems allow the interaction of devices and objects about use-borders away. For our work an adaptive operating surface for the CPS controlling center is to be developed importantly.

Hervas et. al [2] describes the necessary of user interface adaptation to offer personalized information to the user. The kind of user, display and associated visualization requirements may change while handling with the control center for cyber-physical systems. By representing context information, the environment will be able to react to situation changes and determinate the services to be displayed.

Figure 1. Adaptive visual control for multi-devices

Figure 1 shows an abstract of a potential visual robot-control in a smart home scenario. It provides adaptive visualization on connected devices, including smartphones, tablets and PCs.

The topical trend towards the stronger interlinking aims at mobile devices, like tablets or smartphones. Because of the reduced screen size, in comparison to the customary PC, the visualizations and interactions draughts must be adapted to the hardware specifications of such devices. So the information content varies. In addition to the implementation of an adaptive, multi-device solution, it's necessary to support an adaptive, multi-modal control of the CPS by applications. So the interaction with the system can also vary. At the PC the user can handle with the mouse, on a tablet by finger touch or via voice with smartphones.

The integration of different interaction concepts combined with several devices supported the ubiquitous appendage. In this case, we need on the one hand semantic descriptions for the physical resources, like screen size, computation power or technical interaction capabilities for the execution devices. And on the other hand, applications that describe their functional abilities to react and adapt itself on this contextual information. It is than possible to interact with one App on several devices with different interaction methods deduced from the application and the device description.

This context adaptation shows increasingly a key requirement for mobile and ubiquitous systems for our purpose and will be considered in further work. For a flexible data flow in such a system, we introduce our backend architecture for seamless integration in following section.

Seamless Integration Backend

As mentioned before, the user interaction with the cyber-physical system (CPS) is done by the user over Apps. This section shows the infrastructure of our CPS, see Figure 2. The individual CPS elements, applications as well as sensors and actuators, discover the Semantic Middleware to register and communicate seamlessly with the system. Based on the concept of semantic model descriptions on each layer, it leads us to a highly flexible and distributed system.

Application Layer

Apps are executed on the Application Layer, which can distributed over multiple devices like smartphones, tablets or home servers. This layer is contemplate in first order as an abstract layer, so the interoperability is given over multiple

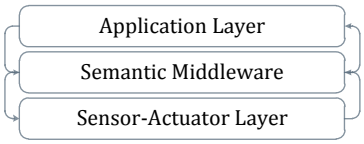

Figure 2. VICCI Backend - Main Structure

heterogeneous devices with different operating systems. The only requirement for every application is to instantiate the App model and provide an interface for data transmission in both directions with the Semantic Middleware. An idea for this connection is presented in the section Conclusion. The App model describe in addition to the user interaction properties the information about input and output data. This means on the one hand the processing and visualization of sensor data in the bottom-up direction and on the other hand the transmission of control instructions to actuators in the top-down direction, that is demonstrated in the section Scenarios. The service part of the App model is adopted from the IOPE approach from OWL-S [6]. But the input and output parameters are described and parametrized in a semantic way, like "get *all* temperature data in the *bathroom*", "get *all* rooms, *that are cooler than 18*°C" or "open *all* windows in the *living room*", which are forwarded to the Semantic Middleware. As mentioned, this description holds an functional part, like "get ...temperature in ..." and a parametrized part "*all*, *living room*", which can be changed during runtime. With this approach the whole system stays highly flexible, because there is no need to adapt the App functionality or change parameters if more sensors and actuators are added to the CPS. The pre- and postcondition of the service, respectively the function of the application, are used for error checking.

Semantic Middleware

The Semantic Middleware (SeMiWa) has the task to acquire, store, interpret, aggregate and route all data flow in the CPS. The acquiration is done over a network interface, which stores all information and knowledge about the individual CPS elements (model instances) in a registrar and opens interfaces for transmitting data to the SeMiWa. The interpretation unit decomposes the semantic annotated IOPE descriptions and annotates in further progress plain data according to device and aggregation models. Another function of the interpretation unit is the knowledge tracking of errors and their solutions adapted from the reaction of the user, if pre- or postcondition fails. The routing unit opens interfaces to Apps and sensors/actuators, and handles the notification of events, based on the IOPE descriptions and the registrar information.

Sensor-Actuator Layer

The Sensor-Actuator Layer is, similar to the Application Layer, designed as an abstract layer, which can also be spread over multiple devices like microcontrollers, robots or home servers. On this layer, low-level sensors and actuators have to implement the so called Semantic Driver, a process that provides a network interface from the hardware devices to the SeMiWa and annotates the plain data against the given sensor/actuator model. If no or an incomplete model for the device is present, SeMiWa tries to annotate it with the right model instance and set the Semantic Driver during runtime.

SCENARIOS

The following section describes two scenarios, which show the data flow through our system. The common situation of both scenarios is the engaged state, in which all components of the cyber-physical system are registered at the Semantic Middleware (SeMiWa) and exchanged their model instances with it.

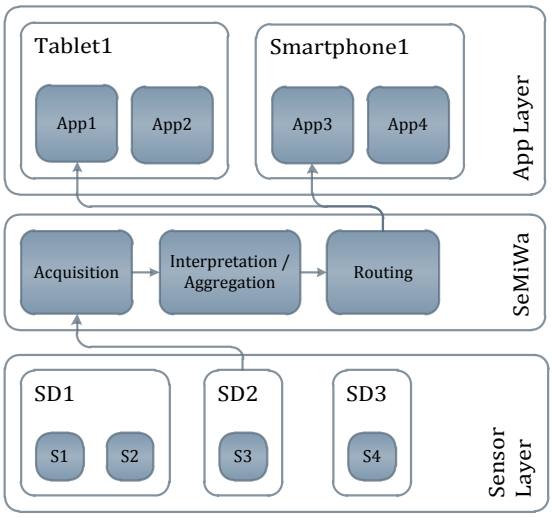

Figure 3. Data flow from sensors

The first scenario Figure 3 shows the data flow bottom-up from the sensor to the user interface via the SeMiWa. The Sensor-Actuator Layer is simplified for this example as pure Sensor Layer. The Sensor Driver *SD2* acquires the plain sensor data from *S3*, and transmit it to the middleware. In due to the registration of *SD2* within SeMiWa, this data can be interpret based on the exchanged model. SeMiWa routes this semantic sensor data to all subscribed Apps, which are interested in this event (*App1* on *Tablet1* & *App3* on *Smartphone1*). These Apps processes the data and visualize it to the user.

The second scenario Figure 4 shows the data flow top-down from one App to one actuator. The numbers in the figure symbolize the order of the ongoing steps. *App1* sends a semantically annotated and parametrized control instruction "open *all* windows in the *living* room", to SeMiWa. After the interpretation of this message, a constraint error is detected based on the precondition "don't open windows in *living* room, if the heatings in *living* room are opened". The user is notified about this error in *App1* and decide to use *App2* to trim off the heatings (*SD2*). The system tracks this decision based on the error, the involved Apps in the right order and the instructions which solved the error. After dissolving of the error, the user uses *App1* again to "open *all* windows in the *living*

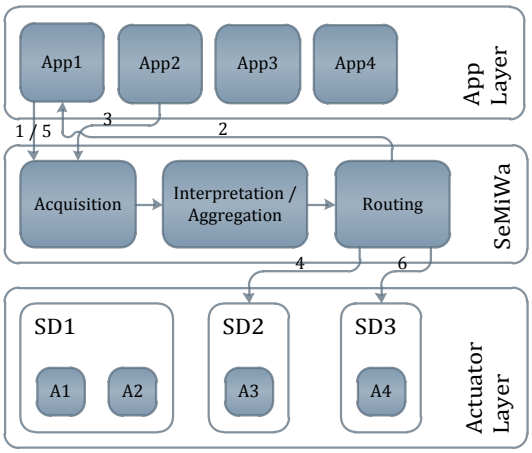

Figure 4. Control flow to actuators

room". This message can now be forwarded to *SD3* without any problems. The system can now assist the user with recommendations, based on previously made decisions if the same error chain reoccurs.

CONCLUSION

We introduced our vision about an interactive control center for cyber-physical systems. This means on the one hand a highly adaptive user interface, which can be spread by Apps on multiple devices based on the underlying model descriptions.

On the other hand, we presented our backend architecture concept. This Semantic Middleware (SeMiWa) helps us to develop a highly flexible and robust system, based on semantic model descriptions on each layer. The input and output descriptions of Apps and the Semantic Drivers stay flexible according to the composed statements, like "close all windows in all rooms". So it is unnecessary, if windows are removed or added to the cyber-physical system. The pre- and postcondition descriptions help us to detect errors and provides a way for knowledge-tracking of the user-made solutions.

As future work, we are focus on the challenging problem to create applicable semantic models for our issues and provide applicable user interfaces on the participating devices. Dashboards can be a good candidate to centralize this large amounts from distributed information, in spite of restricted representation possibilities. Further more an adaptive interaction concept for multi-user- and multi-device-dashboards has to developed for enabling the interoperability and heterogeneity of all participating devices. For the early prototypes, we are going to use technologies like OSGi[1] and Soprano[2] for the development of SeMiWa to get an efficient, and probably realtime, runtime system with lifecycle management. For the App prototypes, we use an Android[3] Smartphone with an UPnP connector for in-house communication and XMPP for WAN communication outside the same subnet. This leads to a flexible, distributed system and interoperability on each layer with all devices [10, 3].

[1] http://www.osgi.org/

[2] http://soprano.sourceforge.net/

[3] http://www.android.com/

ACKNOWLEDGEMENTS

This work is funded under reference ESF-100098171 by means of the European Social Fund (ESF) and the German Free State of Saxony.

REFERENCES

1. Fides-Valero, ., Freddi, M., Furfari, F., and Tazari, M.-R. The persona framework for supporting context-awareness in open distributed systems. In *Ambient Intelligence*, E. Aarts, J. Crowley, B. de Ruyter, H. Gerhuser, A. Pflaum, J. Schmidt, and R. Wichert, Eds., vol. 5355 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2008, 91–108. 10.1007/978-3-540-89617-37.
2. Hervas, R., R., and Bravo, J. Towards the ubiquitous visualization: Adaptive user-interfaces based on the semantic web. *Interacting with Computers 23* (2011), 40–56.
3. Horng, M.-f., and Chen, Y.-t. A new approach based on XMPP and OSGi technology to home automation on Web. *2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM)* (Oct. 2010), 487–490.
4. Issarny, V., Caporuscio, M., and Georgantas, N. A Perspective on the Future of Middleware-based Software Engineering. In *Workshop on the Future of Software Engineering : FOSE 2007* (Minneapolis, United States, 2007), 244–258.
5. Jaroucheh, Z., Liu, X., and Smith, S. A perspective on middleware-oriented context-aware pervasive systems. In *2009 33rd Annual IEEE International Computer Software and Applications Conference*, vol. 2. IEEE Computer Society Press, 2009, 249–254.
6. Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N., and Sycara, K. Bringing semantics to web services: The owl-s approach. In *Semantic Web Services and Web Process Composition*, J. Cardoso and A. Sheth, Eds., vol. 3387 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2005, 26–42.
7. Muro, M., Amoretti, M., Zanichelli, F., and Conte, G. Towards a Flexible Middleware for Context-aware Pervasive and Wearable Systems. In *Engineering* (2010).
8. Poovendran, R. Cyber-physical systems: Close encounters between two parallel worlds. *Proceedings of the IEEE 98*, 8 (aug. 2010), 1363 –1366.
9. Suo, Y., and Shi, Y. Towards initiative smart space model. In *Pervasive Computing and Applications, 2008. ICPCA 2008. Third International Conference on*, vol. 2 (oct. 2008), 747 –752.
10. Suo, Y., and Shi, Y. SSCP: An OSGi-based communication portal for Smart Space. *2009 Joint Conferences on Pervasive Computing JCPC*, 2008 (2009), 309–314.
11. VICCI Research Group. Visual and interactive cyber-physical systems control and integration, 2012. `http://vicci.inf.tu-dresden.de/`.
12. Weiser, M. The computer for the 21st century. *Scientific American 265*, 3 (January 1991), 66–75.

A Semantic Dashboard Description Language for a Process-oriented Dashboard Design Methodology

Maximilien Kintz
Fraunhofer IAO
Nobelstraße 12
70569 Stuttgart Germany
+49 711 970-2182
maximilien.kintz@iao.fraunhofer.de

ABSTRACT
Monitoring and controlling business processes is a challenging task: different data sources need to be combined, appropriate visualizations need to be defined to observe certain goals or Key Performance Indicators (KPI). In this paper, a new semantic dashboard description language used in a process-oriented dashboard design methodology is introduced, to help users focus more on business processes and actual goals and less on technical aspects of monitoring and controlling infrastructure. A reference implementation is described and plans for future improvements of the methodology are introduced.

Keywords
Dashboards, business processes, monitoring, controlling, XML

INTRODUCTION
The monitoring and controlling of business processes places users in front of difficult challenges: multiple and sometimes incompatible data sources have to be integrated, specific needs for the monitoring of precise goals or controlling possibilities of certain process definition values require complex and hard to use Business Intelligence (BI) solutions usually targeted at experts in information analysis, not in the particular business domain being monitored.

As the importance of BI continuously increases (as stated in [9]), to help solve these difficulties, we propose a new process-oriented dashboard design methodology, part of a larger process monitoring methodology and relying on a semantic XML-based dashboard description language.

The remained of this paper is organized as follows: first, the state of the art in dashboard design and usage, related work in process and model driven design methodologies, as well as description languages for semantic user interfaces are presented. Then, the new process-oriented monitoring and dashboard design methodologies are presented. The dashboard description language and its semantic characteristics are then described in detail. A prototype reference implementation is introduced and first results are assessed. Finally, future work is presented with conclusions in the last section.

STATE OF THE ART AND RELATED WORK
Dashboards are nowadays widely used for monitoring and analysis of business processes. Numerous companies such as IBM [14], SAP [15], Tableau Software [16] or TIBCO Spotfire [17], to name a few well-known vendors, offer complete BI or information visualization solutions.

Rules and best practices for the design and use of dashboards are also already widely investigated: from a conception and use point of view, Eckerson proposes in [11] detailed advice. From an information visualization point of view, Few, in [12], gives precise and clear guidelines to follow in order to create dashboards that are easy and efficient to use.

A model-driven design method using BPMN process models to derive dashboards has also already been proposed in [7]. However, this method required information visualization knowledge from its users in order to create efficient designs. The method presented in this paper tries to include this knowledge in the "intelligence" of the dashboard generation and to automate as much as possible. Furthermore, the generation of a full monitoring infrastructure was not foreseen, and possibilities of process controlling supported by our approach were not investigated.

To efficiently describe the dashboards generated by the methodology presented in this paper, a new dashboard description language is introduced. This language helps describing files containing information on graphical visualization, datasets, interaction and controlling elements, alerting and process semantic. Already existing languages offering similar but more limited capabilities (as the semantic and controlling aspects are often forgotten and a higher importance is given to styling and layout) include Vizql [18] derived from the Polaris [23] framework, nowadays integrated in a different form in the Tableau Software solutions and unfortunately proprietary and closed. Standardized serializations for charts in XML

M. Kintz: A Semantic Dashboard Description Language for a Process-oriented Dashboard Design Methodology.
Proc. of 2nd International Workshop on Model-based Interactive Ubiquitous System 2012,
Copenhagen, Denmark, June 25, 2012, http://ceur-ws.org/Vol-947

already exist, such as ChartsML [22] used in a Firefox [25] browser extension or in the chart design tool suite FusionCharts [24]. Other graphical user interface description languages such as Adobe Flex [19] are related. Flex focuses on interactions and allows using XML to design charts but is limited as data sources are concerned. Scalable Vector Graphics (SVG) [20] is mighty but too generic for the use case described in this paper. All these languages tend to be very generic and thus too complex to be easily managed by non-specialists users, and do not offer a sufficient level of semantic to be fully appropriate for our process monitoring and controlling use-case.

PROCESS-ORIENTED SYSTEM MONITORING AND CONTROLLING

The process-oriented dashboard design methodology presented in this article extends a full business process-model-driven monitoring methodology and infrastructure currently in development but already largely implemented and described in [3] and [4] and summarized on Figure 1.

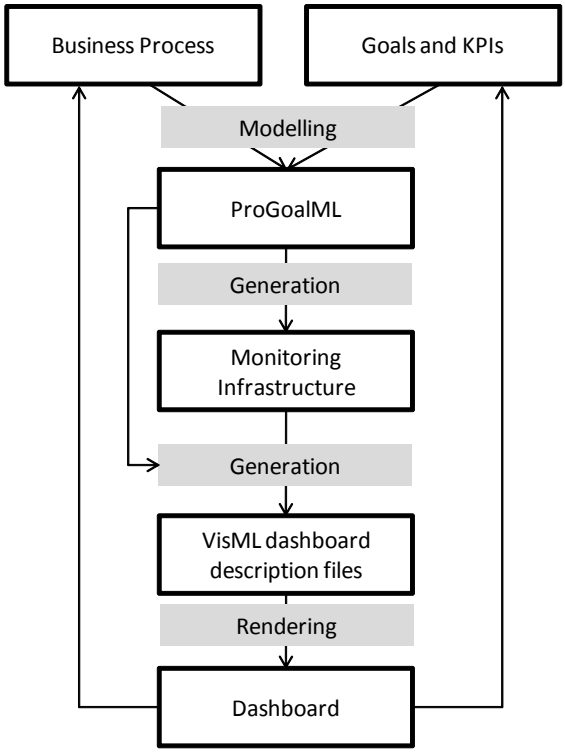

Figure 1: Overview of the process monitoring methodology

Overview of the a.pro Process Monitoring Methodology

The first step of the method involves the users of a system to be monitored defining goals and KPIs, and modeling the business process they want to observe using the BPMN [6] notation. In the process model, possible control points for further process controlling are also indicated. This information, including which goals and KPIs are of interest to which users (i.e. a role concept) is described and stored in an XML file in the specifically created format *ProGoalML*. As this modeling and description operation can be complex, it is performed by an expert, using user input.

Based on this XML file, a whole monitoring infrastructure is then automatically set-up, and code stubs to be integrated in the running process engine in order to send data to a real-time monitoring service using a CEP engine (using the methodology described in [8]) and to a data warehouse for archiving are generated, to be integrated by the responsible IT department in the system.

When this infrastructure has been set up, the dashboard generation phase can begin.

A business-process-model-driven dashboard design methodology

Taking as input the ProGoalML file containing information on the process model, roles and KPIs, as well as access to the CEP engine for real-time information, to the data warehouse for historical information and to control stubs web services if controlling of the process is wanted and allowed by the process model, the VisML generation is started.

The first VisML file generation process includes the following steps:

1. Matching goals to data types (as is already done in the process monitoring methodology for the creation of the CEP engine and data warehouse);
2. Matching data types to visualizations (cf. Figure 2), answering typical questions and following processes in a way similar to that defined in [21] and using best practices for the choice and styling of charts described in information visualization literature, for example in [1]. To allow for a flexible and easy update of the matching algorithm, the matching configuration is stored in an XML file containing pairs of ProGoalML goals descriptors and of VisML charts descriptors;
3. Matching visualizations to the correct data source (CEP engine Web service interface for real time data, data warehouse for historical data) and defining appropriate options for drill-down and other interactions;
4. Defining proper alerting and controlling commands, using goals defined in the ProGoalML file and information from the process model;
5. Generating a basic dashboard layout for each role, by enabling only the relevant visualizations for a specific role, and sorting them according to both priorities defined in the roles and views model accompanying the ProGoalML and to semantic criteria (i.e. placing visualizations related to the same process steps next to each other).

As usual with model-driven graphical interface generation [5] (dashboards being here a specific type of user interface),

the mapping described in steps 1 to 3 is of particular importance.

Once one or more (depending on the number of roles and views specified) VisML files have been generated, these can be loaded in a compatible dashboarding engine. There, the users can use and modify them (by hiding visualization, switching places or changing sizes, disabling some alerts, etc.) as necessary, and thus recreate customized VisML files, should they not be entirely satisfied with the automatically generated ones.

Data type	*Visualization*
Composition, categories	Bar chart
Comparison over time, distribution	Line chart
Single value	Number, possibly sparkline
Difference actual value vs. objective	Bullet graph

Figure 2: Mapping data type to visualization

Although the customization is possible at any time, a normal use case would consist on a limited number of iterations for the modifications of the VisML files, after which monitoring and controlling tasks would simply be performed as in traditional BI and monitoring solutions.

The components of the process-model-driven dashboard design methodology and their interactions are presented on Figure 3.

A SEMANTIC DASHBOARD DESCRIPTION LANGUAGE

The dashboards created with the new dashboard design method have specificities such as semantic information and controlling possibilities which make it necessary to describe them with a new language, different than the ones currently used by dashboard solutions vendors. We call this new language *VisML* for *visualization markup language*, as it could theoretically be used to describe any kind of visualization, not only dashboards.

Design goals for the VisML language were to be highly semantic (focusing on describing the meaning of elements and leaving technical aspects such as layout or styling to the dashboard rendering engine, thus making it possible to adapt the same VisML dashboard to a desktop, tablet or smartphone layout, for example), easily human readable, and overall as simple as possible (i.e. limited to the information that is absolutely necessary to render the dashboard).

The overall structure of a dashboard file described in the new semantic dashboard description language is presented in the next paragraphs, an example is shown on Figure 4.

Overview

A dashboard file should contain the information needed by a software tool to render a graphic view of the dashboard as specified by the user, and allow for required interactions such as drill-down or controlling. To achieve this, it appeared necessary to include generic information on the dashboard, an exhaustive list of all visualizations that compose the dashboard, some specific conditions for monitoring use cases such as alerting information, and references to the data to be displayed.

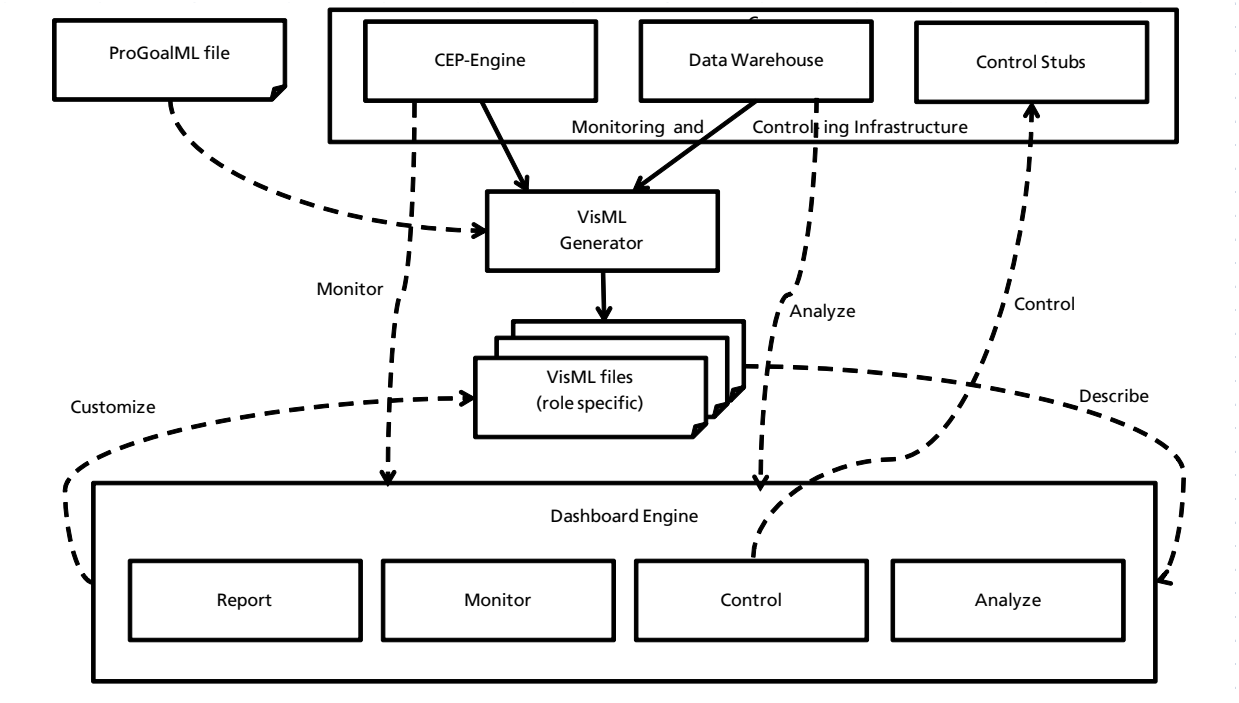

Figure 3: The components of the dashboard design methodology

We use XML [10] to create documents that are easily human and machine readable.

The structure of a dashboard file is therefore specified as an XML file containing a top element *vismL*, itself containing exactly four sub-elements:

1. The element *meta*, containing generic information such as title, author, etc.
2. The element *dashboard*, containing a collection of *visualization* elements, each describing a specific visual part of the dashboard.
3. The element *alerting*, containing a series of alert elements, specifying conditions to be monitored and actions to be triggered when a condition is reached.
4. The element *data*, containing a series of data sources and sets.

In the following paragraphs, we describe the role and structure of each of these four sub-elements.

The Element META

The element *meta* (for metadata) contains generic information describing the dashboard. The element is composed of two sub-elements: *description* and *semantic*. The sub-element *description* is composed of a mandatory element *title*, indicating a title given to the dashboard (used to be display as a window title, for storage, etc.), zero or more *author* elements, used to store the names of the dashboard authors, and two optional *datetime* and *comment* elements, used to store a modification date and time of the dashboard structure (for versioning) and additional comments.

The element semantic is described in a specific section.

The Elements DASHBOARD and VISUALIZATION

The main element of a dashboard description document is the element *dashboard*. It represents the part of the dashboard that must be visually rendered by the supporting tool and presented to the user.

A dashboard consists in a collection of *visualization* elements. Each visualization represents a graphical display of information, theoretically in any possible form. However, the dashboard description language focuses on the description of business dashboards and therefore privileges those displays that are typical in such dashboards, such as line graphs, bar graphs, tables and numbers with sparklines [2], or bullet graphs [6].

A visualization is defined by its attributes *category* and *type*. The possible *category* values are *chart*, *graph*, *map*, *diagram*, *table*, *text* and *other*. This list is a slightly adapted version of the classification of information graphics introduced in (Harris, 1999). *Text* has been added because simple textual messages or numbers are often useful on dashboards; other ensures the completeness of the specification.

Other characteristics of a visualization are a *dataset* (see related section), a *title* and optional *description*, an element *interaction* that contains a list of possible interaction capabilities such as *zoom*, *pan* or *drill-down*, *style* and *semantic* information.

```
<?xml version="1.0" encoding="UTF-8"?>

<vismL version="" xmlns="...">

<meta>
          <title>Title of the dashoard</title>
          <author>Author name</author>
...
</meta>

<dashboard>
<visualization dataset="set1" id="vis1" category="chart"
type="line">
          <title>Line chart title</title>
          <interaction type="click">
          <scope>...</scope>
          </interaction>
          </visualization>
...
</dashboard>

<alerting>
<alert id="1" dataset="set1"
<condition><![CDATA[saving > 10]]></condition>
<mailNotification frequency="EVERY 30minutes">
<email>mail@example.org</email>
<message>This is the e-mail alert message.</message>
</mailNotification>
</alert>
...
</alerting>

<data>

<datasources>
<database id="source1">…</database>
...
</datasources>

<datasets>
<dataset datasource="source1" id="set1" format="..."
type="...">
<semantic><relation id="..." type="..."/></semantic>
<query><![CDATA[SELECT X AS NAME, COUNT(COST) AS VALUE
FROM costs WHERE COST <= 499;]]></query>
</dataset>
...
</datasets>
...
</data>

</vismL>
```

Figure 4: Excerpt from an example VisML dashboard description file

The Element ALERTING

It is possible to define alerting conditions and messages in a VisML file. When a certain dataset meets a condition, the specified corresponding alert message can pop-up in the dashboard window or can be sent by e-mail to a list of recipients. Other actions can of course be imagined (sending an SMS or even impacting a system), as long as they can properly be supported by the monitoring and controlling engines.

An *alert* element typically consists of a *condition* referring to a dataset and one or more notifications, each with their own frequency and messages.

The Element SEMANTIC

A specify of the VisML dashboard description language it its ability to not only describe the appearance of

visualizations but also their meaning and links to actual business processes or process steps, which can easily be done since the dashboard design method is business process model-driven.

For each dashboard in the element *description* and for each individual data set, relations to process steps identified by their ID as specified in the business-model part of the ProGoalML file can be specified. The *semantic* simply consists of one or more *relation* elements, each having three attributes: *id* for the process-step being referenced, *priority* for differentiating relations when several are attributed to the same visualization, and *type*.

This semantic information is then used by the dashboard rendering engine to help sort and place the visualizations in meaningful ways, and to better manage links to the actual process steps for use-cases such as controlling of the business process.

The Elements DATA, DATASOURCE and DATASET

The element *data* contains two main sub-elements: *datasources* and *datasets*.

The element *datasources* contains a list of sources of dashboard data. A data source is a reference (for example, the connection parameters) to a specific database, a Web service delivering data, or possibly a document containing data in a format supported by the dashboarding engine (this use-case is not supported in the current implementation).

The element *datasets* contains a series of sets of dashboard data, derived from the data sources specified before. A *dataset* is a specific reference to a data source table, a query on a database or a list of values to be retrieved from a Web service. A dataset can also be defined as a *meta-dataset*, i.e. as the resulting combination of other *dataset* elements already defined in the VisML file. The *dataset* also contains lists of labels to be mapped to value categories and later displayed as labels on the visualization, for example on the axis of graphs. Each dataset may contain several relations to process models or steps defined with the element *semantic*.

Linking VisML with common BI solutions

As the VisML dashboard description files are simple XML documents, they can be processed, for example with XSLT style sheets, to be transferred into dashboard files compatible with other already existing BI solutions. Necessary conditions to that end are that the vendor-specific dashboard documents do not require more information than is available in the VisML file, that the vendor-specific language is documented and the proper interpreter is implemented.

EXAMPLE IMPLEMENTATION FOR INSURANCE CLAIMS MANAGEMENT

Large parts of the a.pro process monitoring methodology and of the dashboard design methods described in this paper have already been implemented. The VisML semantic dashboard description language is currently in productive use in a lightweight dashboard for the monitoring of an insurance claims management tool, easily allowing users to simply edit and adapt their dashboards from a Web browser-based full Javascript interface. An example VisML dashboard is presented on Figure 5.

Feedback gathered from the use of the lightweight VisML dashboards show that the simplicity of the language is an important aspect, as it allows users to immediately adapt the dashboard to their needs if the generated version wasn't satisfying. Although interactions are supported by the

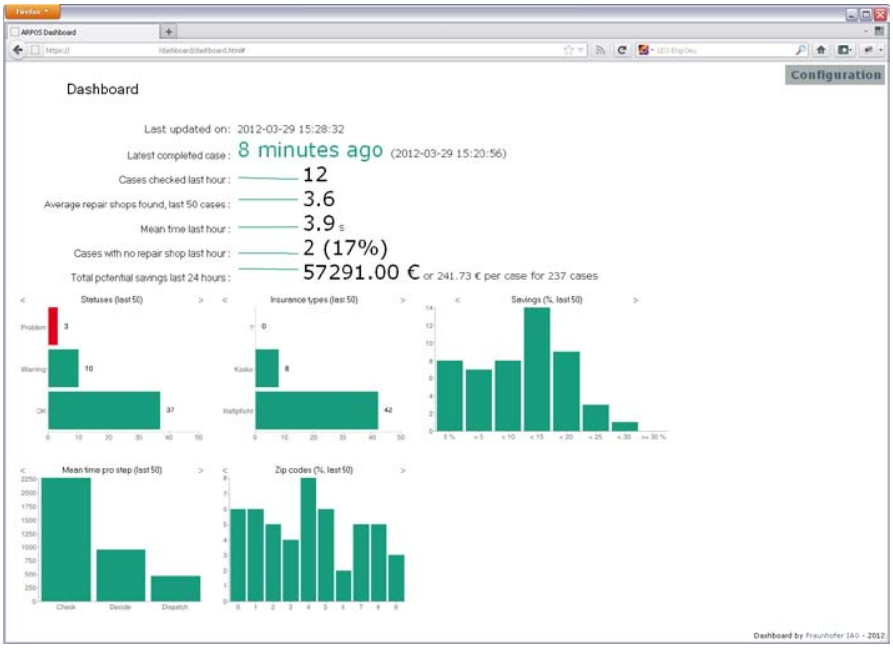

Figure 5: An example VisML dashboard

language specification, they do not for all use-cases need to be supported by the rendering engine, as users of dashboards for monitoring purposes do not necessarily always need to perform advanced exploratory data analysis tasks.

The full automatic generation of the dashboards and the use of control points to actively impact the business process being monitored still needs to be implemented and tested.

CONCLUSION

To help overcome known challenges of business process monitoring, such as the difficulty to built appropriate dashboards from complex data sources to best monitor given goals, a new semantic dashboard description language used in a process-oriented dashboard design methodology was introduced. This methodology is part of a larger business-process monitoring and controlling methodology. Large parts of these solutions have already been implemented and successfully used in production.

The process monitoring solution we presented offers several characteristics of ubiquitous computing as defined in [26]: it allows for the easy and seamless interaction and monitoring of complex and multiple IT systems, it helps the users forget about technical details and focus on business processes and goals, and as the VisML dashboard description language we introduced emphasizes semantics over styling, it can be used to render adapted views of the dashboards on many different systems, such as single or multi-screen desktops, tablets or even smartphones.

Future work focuses on the improved fully automated generation of dashboards and on the possibilities of active business process controlling using the monitoring dashboards as configuration panels.

REFERENCES

1. S. Few. *Effectively Communicating Numbers: Selecting the Best Means and Manner of Display*, 2005. http://www.rit.edu/cla/cpsi/SRResources/Effectively%20Communicating%20Numbers.pdf
2. E. R. Tufte. *Beautiful Evidence. Graphics Press, 2006.*
3. F. Koetter and M. Kochanowski. Goal-Oriented Model-Driven Business Process Monitoring using ProGoalML. In *Proceedings of the 15th International Conference on Business Information Systems (BIS 2012*) (in press), 2012.
4. F. Koetter, A. Weisbecker and T. Renner. Business Process Optimization in Cross-Company Service Networks – Architecture and Maturity Model. In *Proceedings of the 2012 Annual SRII Global Conference*, 2012.
5. T. Schlegel.; M. Raschke.; M. Knittig.; A. Dridiger.; S. Wokusch. and C. Taras. Evaluation of Current User Interface Generator Frameworks for Graphical Interactive Systems. In *Proceedings of the IADIS International Conference Interfaces and Human Computer Interaction 2010*, 2010.
6. S. Few. *Bullet Graph Design Specification*, 2010. http://www.perceptualedge.com/articles/misc/Bullet_Graph_Design_Spec.pdf
7. P. Chowdhary, T. Palpanas, F. Pinel, S.-K. Chen, and F. Y. Wu. Model driven dashboards for business performance reporting. In *Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC'06)*, 2006.
8. T. Schlegel; S. Dusch and K. Vidackovic. Interaction- and Event-Based Management of Processes in Service-Oriented Infrastructures. In *Proceedings of the 6th I*PROMS virtual conference*, 2010.
9. J. Hagerty, R. L. Sallam, and J. Richardson. *Magic quadrant for business intelligence platforms*. 2012.
10. W3C. *Extensible Markup Language (XML) 1.0 (Fifth Edition)*, 2008.
11. W.W. Eckerson. *Performance Dashboards: Measuring, Monitoring, and Managing Your Business*. John Wiley & Sons, Inc., 2011.
12. S. Few. *Information Dashboard Design*. O'Reily Media, Inc., 2006.
13. OMG. *Business Process Model and Notation (BPMN) Version 2.0*, 2009.
14. IBM Business Analytics. http://www-142.ibm.com/software/products/us/en/category/SWQ00
15. SAP Businessobjects Business Intelligence Solutions. http://www.sap.com/solutions/sapbusinessobjects/large/business-intelligence/index.epx
16. Tableau Software. http://www.tableausoftware.com/
17. TIBCO Spotfire. http://spotfire.tibco.com/
18. P. Hanrahan. Vizql: A language for query, analysis and visualization. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, 2006.
19. Adobe Flex. http://www.adobe.com/products/flex.html
20. W3C. SVG Working Group. http://www.w3.org/Graphics/SVG/
21. X.J. Li; T. Schlegel; M. Rotard and T. Ertl. A Model-Based Graphical User-Interface for Process Control Systems in Manufacturing. In *Proceedings of the Intelligent Production Machines and Systems - 2nd I*PROMS Virtual International Conference*, 2006.
22. T. Saito. *ChartML*. 2008. http://www.onsaito.com/csdv/chartMLindex.xhtml
23. C. Stolte; D. Tang. and P. Hanrahan. Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases. In *IEEE Transactions on Visualization and Computer Graphics* 8, 2002.
24. FusionCharts. *Charts XML Reference*. http://docs.fusioncharts.com/free/
25. Mozilla Firefox. http://www.mozilla.org/en-US/firefox/new/
26. M. Weiser. Hot Topics: Ubiquitous Computing. In *IEEE Computer*, October 1999.

Test Modeling for Context-aware Ubiquitous Applications with Feature Petri Nets

Georg Püschel
Technische Universität Dresden
Georg.Pueschel1@mailbox.tu-dresden.de

Ronny Seiger
Technische Universität Dresden
Ronny.Seiger@tu-dresden.de

Thomas Schlegel
Technische Universität Dresden
Thomas.Schlegel@tu-dresden.de

ABSTRACT
Applications for ubiquitous systems have to be designed to run in contextual environments and on a multitude of software and hardware platforms. To assure their quality in an adequate subset of these static as well as dynamic configurations, variability modeling and model-based testing can be used. In this paper, we present an approach applying Model-based Testing (MBT) and Dynamic Feature Petri Nets (DFPN) to define a test model from which an extensive test suite can be derived. We argue that our method is capable of efficiently modeling context-aware application behavior for test purposes.

Author Keywords
Model-based Testing, Petri Nets, Ubiquitous Systems, Context, Features, Mobile Applications

ACM Classification Keywords
D.2.5 Software Engineering: Testing and Debugging

INTRODUCTION

Mobile and ubiquitous applications are designed and developed to be executed on a multitude of heterogeneous target platforms, i. e., supporting a large number of software and hardware configurations. In addition, they have to adapt to changes within their environment based on the current context model describing "external data, that may influence the application"[1], e.g., location or connectivity information. To support platform independence, software and programming interfaces abstract from platform specific properties. However, the resulting behavior of a system may still differ due to minor implementation differences. Hence, we have to test this specific software within an adequate set of predefined configurations.

The second issue —changing contexts— occurs at runtime. The *system under test* (SUT), depends, e. g., on geospatial data (GPS), adapts to connectivity problems, or changes its graphical interface according to the device's current orientation. In order to test this behavior, we have to keep in mind that these *dynamic* adaptations can occur at almost arbitrary points in time.

To deal with those problems, we designed a workflow for generating test cases, which is depicted in Fig. 1. First, we make use of *feature models* [3], which are a widely-used concept to define commonality and variation in systems, especially in Software Product Lines (SPLs). The static variability of

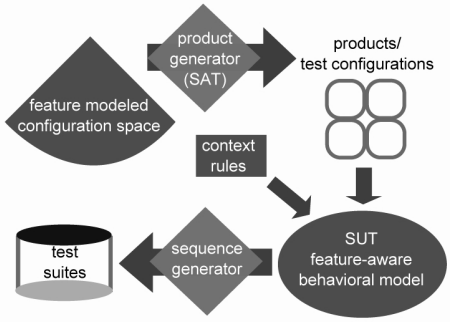

Figure 1. Workflow of test case generation.

configurations based on software and hardware platform differences can be defined by using feature models spanning a test configuration space. From this space, configurations can either be selected specifically for testing or they can be verified against the feature models' constraints.

Second, we apply *model-based testing* (MBT), which is a means for "the automation of black box tests" [10]. In MBT, models are used to specify the communication between a test environment and the SUT —considered to be a black box— via its interfaces and to generate valid test cases. The main advantages of MBT include a reduction of redundancy, a measurable coverage, and a traceability among artefacts. Thus, we create a test model based on Dynamic Feature Petri Nets (DFPN) to define the SUT's behavior under all possible configurations, and to derive a subset of this model from *context rules*. These context rules associate feature activations or deactivations with actions that have to be executed in order to enable the (de-)activation in the black box SUT. Combining the test model, context rules, and feature models, a generator derives one test suite for each of the application's static configurations.

Overview

The rest of this document is structured as follows: first, we design the features models of an exemplary SUT as well as its Petri net-based behavioral test models. Subsequently, it is shown how parts of our test model are derived from context rules and how test cases are generated. Afterwards, we briefly present an editing tool which implements our approach. In the end, we discuss related work, conclude our own contributions, and outline future research activities.

G. Püschel, R. Seiger, T. Schlegel: Test Modeling for Context-aware Ubiquitous Applications with Feature Petri Nets.
Proc. of 2nd International Workshop on Model-based Interactive Ubiquitous System 2012,
Copenhagen, Denmark, June 25, 2012, http://ceur-ws.org/Vol-947

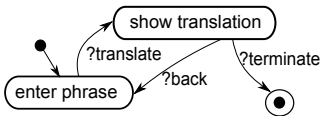

Figure 2. State machine for the TransApp example.

MODELING APPROACH

In this section, we present our modeling strategy. To illustrate our approach, we introduce a brief example from the mobile application domain in the beginning. Fig. 2 shows the state machine for our prototype application called "TransApp", a minimalist translation agent. It consists of two states, the first one enabling the user to enter a sentence and letting the service backend translate it. Second, the user can either terminate the application or return back to translating another phrase. The target language is thereby determined automatically using the system's `Localization` feature and translation is done by the `Translation` feature via local or Internet dictionary look-up. In the following, we systematically build and combine models to define all information necessary for the test case generation.

Configuration Space Feature Models

As discussed initially, one of the most important tasks in test modeling of context-aware systems is to manage variability. In SPL research, this problem has been approached by using feature models, which can also be extended to dependent feature models with the help of relational semantics (more detailed in [7]). Fig. 3 shows the usage of two dependent feature models for TransApp. The feature models are presented as *feature trees*. The upper tree defines the variability of the SUT itself. Apart from the common `core` feature, which contains entities shared among all instances, two abstract features must be selected in all valid products. `Localization` is used to provide an automatic selection of the target language for translation and may either be implemented by using a GPS sensor or by using cellular network data. These specializations are mutually exclusive, i. e., only one of them will be used for localization (marked by an arc between the specialization relationships). For `Translation`, two mutually exclusive options exist as well. Either the system is connected to the Internet and, therefore, an online database is used for looking up the translation, or a local dictionary is consulted.

The lower feature model in Fig. 3 defines the variability space of the underlying system, i. e., of the test environment. The platform consists of a GPS and of a cell phone feature. Internet connection is optional (marked by an empty circle ◯). By defining features as non-optional (standard relationship line with no markings), we do not state that they are mandatory, but that we exclude them from our test configuration set.

Both feature trees are composed by *required* constraints (marked by arrows ↗). For instance, `GPS Localization` requires a `GPS` feature in the underlying system environment. Later on, these relations enable the generator to derive configuration steps leading to a specific test setting.

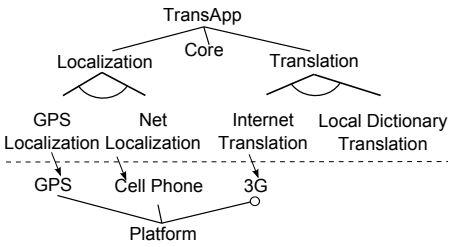

Figure 3. Dependent feature models for TransApp.

Based on the information from the feature models, a set of valid configurations can be generated, e. g.,

$$\{GPS\ Local., GPS, Local\ Dict.\ Translation\}$$

. We can use each configuration as a starting point for a behavioral model defining which features are bound to the system's initial state.

Feature Petri Nets and Test Modeling

Modeling behavior in variable systems can be done by using specific models that define a common behavior space for all selections of features possible. However, many of these metamodels, e. g., modality enriched activity diagrams, have no semantics for feature control. Hence, Muschevici et al. proposed in [5] a Petri net-based model, in which both feature dependent behavior (*Feature Petri Nets*, FPN) and feature controlling behavior at runtime (*Dynamic Feature Petri Nets*, DFPN) can be expressed. While the exact details can be taken from their publications, we give an informal but intuitive definition here and show an exemplary illustration of a DFPN for our test model application in Fig. 4.

First of all, DFPNs are based on basic Petri nets. *Places* (visualized as circles) are partial states which can be connected with *transitions* (black bars) by directed *arcs*. Arcs' starting and ending points have pairwise disjunct types so that they never connect two places or two transitions. A *marking* can be created by putting *tokens* (black dots) into places. A transition is *activated* when all places, which have an outgoing connection with this transition, are marked with tokens. The state of a Petri net can be changed by removing these "incoming" tokens and putting new tokens in all places directly connected by the outgoing arcs from the activated transition. A *trace* is a sequence of states. As multiple places can be marked in the same state, Petri nets can be used to model distributed and parallel processes.

DFPNs adapt the Petri net notion and extend it by marking transitions with *application conditions* and *update expressions*. In Fig. 4, transitions are annotated with these elements in the following way:

$$\frac{\text{application condition}}{\text{update expression}}$$

. An application condition restricts the activation (firing) of transitions to the set of bound (activated) features. At system start, a predefined feature selection is used for initialization. The syntax of a condition follows the grammar

"$\varphi ::= a \mid \varphi \wedge \varphi \mid \neg\varphi$, with $a \in F$" [5]

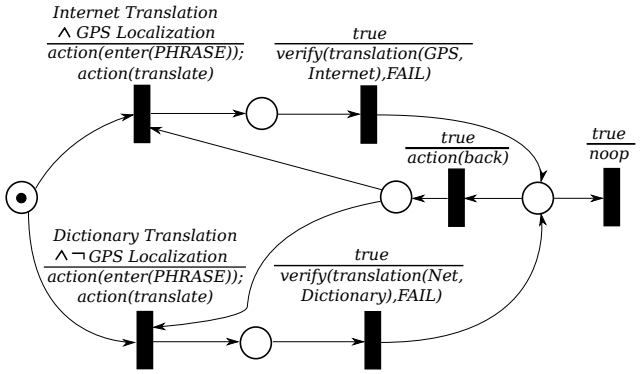

Figure 4. Dynamic Feature Petri Net of our test model.

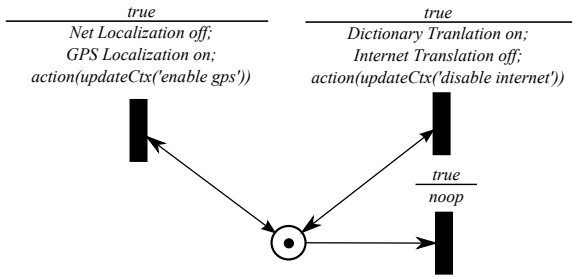

Figure 5. Extract of the context feature model.

where F is the active feature set. We redefined this for reasons of convenience to

$$\varphi ::= a \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid \neg\varphi \mid true$$

. In Fig. 4, for instance at the beginning, the decision depends on the activation state of the GPS feature, which was determined in advance.

Furthermore, update expressions control the (de-)activation of a feature a (`on/off`). Muschevici et al. defined their grammar as:

$$\text{“}u ::= \texttt{noop} \mid a\ \texttt{on} \mid a\ \texttt{off} \mid u; u\text{”}\ [5]$$

. We added two statements, namely `action` and `verify`, to this notation and extended it to

$$u' ::= u \mid \texttt{action(x)} \mid \texttt{verify(x,v)} \mid u'; u'$$

where x is a term and v is a verdict $\in \{PASS, FAIL\}$. This allows test modelers to define black box interface interactions with the help of arbitrary message patterns (e. g., send-receive, receive-send), whereas an `action(x)` operation indicates to send data to the SUT and `verify(x,v)` operations receive and immediately verify data from it.

For our example case (cf. Fig. 4), we use update expressions to navigate through the application and to validate that a translation was executed correctly. The verified terms may be interpreted as equivalence classes for the test data. By using DFPNs in combination with our extensions, we are able to model feature activation dependent test cases and feature controlling steps.

Defining Feature Dynamics

As presented in Fig. 4, we did not use `on` and `off` operations directly on features in the test model. However, context is the actual driver for changing feature states with respect to the targeted application and system types. We connect both operations with *context rules*, having the form

$$\varphi(\texttt{on}|\texttt{off}) \Rightarrow (\texttt{action(x)}\,|\texttt{verify(x,v)})$$

and use those to generate and insert a parallel context controlling branch into our test models, as depicted in Fig. 5.

For example, the context rule

$$GPS\ Localization\ \texttt{on} \Rightarrow action('enable\ GPS')$$

would produce the upper left transition. An action performed as a consequence of a feature activation or deactivation must lead to this indirectly controllable operation. For instance, if we perform this test manually, such an operation would contain an instruction for the tester to place the device in a location within GPS satellite range. A more automatic interpretation would be to execute an update over an ontologically modeled context (e. g., by using SPARQL queries).

Using the feature models defined initially, we can derive that it is necessary to deactivate `Net Localization`, because both features exclude each other mutually. The context branch contains exactly one central place, so that in every simulation step only one of its transitions may be activated. Through this parallelism and the feature activation states, the test model and the context branch are able to interact with each other.

Generation of Test Cases

To generate test cases, we have to perform a reachability analysis for the Petri net, i. e., a complete simulation of the combined DFPN consisting of the test model and the context branch to derive all possible traces, each of which corresponding to a specific test case. Additionally, we need to filter direct feature control operations (`on/off`), keeping only actions and verifications with runtime relevance. As the latter ones do not influence the (D)FPN execution semantics, we claim that all provable properties of the Petri net also hold true for our extensions.

TOOL SUPPORT

The implementation of our approach is in ongoing development. The *Mobile Application Test Environment (MATE*[1]*)* prototype is depicted in 6. The editor is based on Eclipse and provides a toolset for DFPN-based test models (**1**), the creation of context rules (**2**), and of feature models (**3**). With the help of these models, specific test cases and classes of test cases can be derived to achieve a large coverage of the feature and test space. MATE is also capable of test suite generation and execution. Additionally, an interface is provided to use technology and platform specific test drivers.

RELATED WORK

Our research is related to and influenced by a broad range of research fields. Ichiro Satoh [9] proposed an emulator-based approach for network migration simulation in mobile

[1] **http://www.quality-mate.org/**

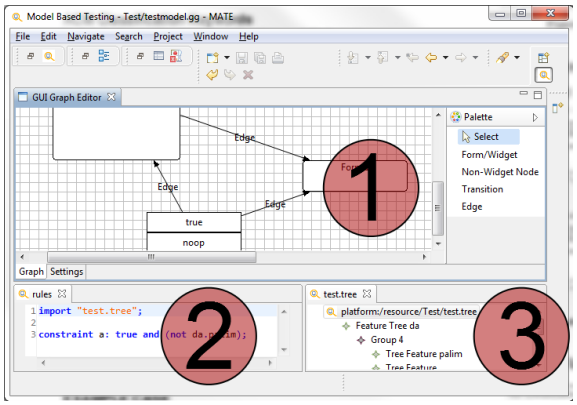

Figure 6. MATE – Mobile Application Test Environment.

computing, which is one aspect of context, while we actually focus on several ones. A more recent work [11] integrated context changes into manually constructed test cases.

Fernandes et al. integrated feature and context modeling in [2] and used specific notations (UbiFEX and Odyssey-FEX). In [12], White and Schmidt proposed using feature models for mobile application design, too. They base their claim on the need for variability management. Similarly, Ridene and Babier manage variability of mobile systems explicitly in test models. For this purpose, they designed a domain specific modeling language [6].

Considering behavioral models for dynamically variable systems, Muscevici et al. compared their DFPN approach to several other Petri net extensions, e. g., self-modifying [8] or reconfigurable [4] Petri nets and stated that they can be mapped to each other.

CONCLUSION AND FUTURE WORK

By applying our approach, it is feasible to link variability in mobile and ubiquitous systems with behavioral test models to generate test suites for different configurations and to change context settings at runtime. To achieve this, we used feature models to define variability of different systems under test and test environment dimensions. With the help of context rules, test engineers can map runtime-dynamic features to test steps necessary to activate or deactivate (bind/unbind) features.

The test model itself is based on Dynamic Feature Petri Nets (DFPN) enriched with a generated parallel branch derived from the information given in the context rules. We claim that this approach provides an expressive means to efficiently create test models for dynamically variable systems, for instance in the ubiquitous and context-aware domain. Compared with related research, we achieve a large and measurable coverage of the configuration space including a variety of context properties, high levels of reusability and automation through modeling, and an extended formalization of test scenarios.

With regard to future work, we intend to evaluate MATE in an industrial case study. In that process, we will measure the effort of creating test models and also its benefit for test projects in comparison to the conventional manual process. Future research problems include coverage constraints that address variability, the introduction of traceability mechanisms, and the design of expressive graphical and textual modeling languages to support test experts.

Acknowledgments

This research has received funding within the projects #100084131 and #100098171 by the European Social Fund (ESF) and the German Federal State of Saxony as well as T-Systems Multimedia Solutions GmbH.

REFERENCES

1. Dalmau, M., Roose, P., and Laplace, S. Context aware adaptable applications - a global approach. *CoRR abs/0909.2090* (2009).
2. Fernandes, P., Werner, C., and Murta, L. Feature modeling for context-aware software product lines. In *Proceedings of the 20th International Conference on Software Engineering & Knowledge Engineering (San Francisco, CA, USA, 2008)* (2008).
3. Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E., and Peterson, A. S. Feature-oriented domain analysis (FODA). Tech. rep., Software Engineering Institute, Carnegie Mellon University, 1990.
4. Llorens, M., and Oliver, J. Structural and dynamic changes in concurrent systems: reconfigurable Petri nets. *Computers, IEEE Transactions on 53*, 9 (2004), 1147–1158.
5. Muschevici, R., Clarke, D., and Proenca, J. Feature petri nets. In *Proceedings of the 14th International Software Product Line Conference (SPLC 2010)* (2010).
6. Ridene, Y., and Barbier, F. A model-driven approach for automating mobile applications testing. In *Proceedings of the 5th European Conference on Software Architecture*, ACM Press (2011), 9.
7. Rosenmüller, M., Siegmund, N., Thüm, T., and Saake, G. Multi-dimensional variability modeling. In *Proceedings of the Workshop on Variability Modelling of Software-intensive Systems (VaMoS)* (2011).
8. Rüdiger, V. Self-modifying nets: A natural extension of Petri nets. In *Automata, Languages and Programming*, vol. 62 of *Lecture Notes in Computer Science*, Springer (1978), 464–476.
9. Satoh, I. A Testing Framework for Mobile Computing Software. *IEEE Transactions on Software Engineering 29*, 12 (2003), 1112–1121.
10. Utting, M. *Practical model-based testing: a tools approach*. Morgan Kaufmann, 2007.
11. Wang, H., and Chan, W. K. Weaving context sensitivity into test suite construction. In *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*, ASE '09, IEEE Computer Society Press (2009), 610–614.
12. White, J., and Schmidt, D. C. Model-driven product-line architectures for mobile devices. *Journal On The Theory Of Ordered Sets And Its Applications* (2008), 9296–9301.


S E S


MMT


TECHNISCHE
UNIVERSITÄT
DRESDEN