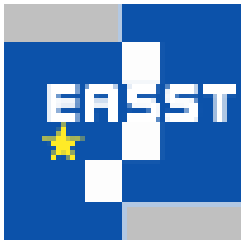
THE EASST NEWSLETTER

Volume 8

June 2004



European Association of Software Science and Technology

EASST **Board:**
Dr. Tiziana Margaria (President),
email : tmargaria@metaframe.de
Prof. Dr. Hartmut Ehrig (Vice-President, ETAPS 2000 organizer),
email : ehrig@cs.tu-berlin.de
Prof. Dr. Herbert Weber (Treasurer),
email : herbert.weber@isst.fhg.de
Dr. Julia Padberg (Secretary),
email : padberg@tzi.de
Prof. Dr. Marie-Claude Gaudel (representative in the ETAPS steering committee),
email : Marie-Claude.Gaudel@lri.fr
Prof. Dr. Egidio Astesiano (representative in the ETAPS steering committee),
email : astes@disi.unige.it
Dr. Michel Wermelinger (column editor),
email : mw@di.fct.unl.pt
Prof. Susanne Graf,
email : Susanne.Graf@imag.fr

Homepage of EASST**:**
http://www.easst.org

Subscription:
EASST NEWSLETTER is distributed among the members of EASST, the *European Association of Software Science and Technology*. If you are not yet a member of EASST, but you wish to receive the EASST NEWSLETTER, then you are kindly invited to become a member! Note, there are **no** membership fees. The application form can be found on the last page.
Or apply online at http://www.easst.org

Editor:
Dr. Julia Padberg
Technische Universität Berlin
FB 13, Sekr. FR 6-1, Franklinstr. 28/29, D-10587 Berlin

E-mail : padberg@cs.tu-berlin.de
URL : http://tfs.cs.tu-berlin.de/~padberg/
Tel : +49-30/314-24165
FAX : +49-30/314-23516

Contents:

Welcome!

Dear EASST Members,

this newsletter´s topic is ETAPS 2004 in Barcelona, a huge success, as you can see from the various reports we have in this issue. First I would like to congratulate the winners of the EASST Award M. Huisman, D. Gurov, C. Sprenger, and G. Chugunov with their contribution *Checking Absence of Illicit Applet Interactions: A Case Study* to FASE 04. You find a summary of this paper in this newsletter.
Then there are two topics of EASST itself, the minutes of the General Assembly that has taken place as usual during FASE and a guideline for the EASST representatives.
Moreover, there is a new Column, THE VISUAL MODELING TECHNIQUES COLUMN, edited by R. Heckel with his contribution *Introducing SegraVis*. Last, but not least THE SOFTWARE ARCHITECTURE COLUMN – now edited by R.Reussner and M. Wermelinger – we have two contributions: *Language Engineering for Model-driven Software Development* by R. Heckel and J. Bézivin and the report on the foundation of the new working group Software Architecture of the German computer science society (GI-AK SoftArch) by R. Reussner.
Then, especially for those that could not attend ETAPS, we report on ETAPS in general, on FASE and on two workshops, namely the INT and the FESCA workshop.
And we end the newsletter announcing FMICS, and as usual with the EASST Flyer and the application form.

So, I am sure this is a worthwhile EASST-Newsletter and I want to thank all colleagues, that have contributed.

Yours sincerely,

Julia Padberg
(EASST-Secretary)

PS: Again, if you have any contribution you would like to have in the forthcoming newsletter, please send it to me.

A Summary of: Checking Absence of Illicit Applet Interactions: A Case Study[1]

Marieke Huisman* and Dilian Gurov**
and Christoph Sprenger[2]* and Gennady Chugunov******
*INRIA Sophia Antipolis, France
**Royal Institute of Technology,Kista, Sweden
***Swiss Federal Institute of Technology, Zurich, Switzerland
****Swedish Institute of Computer Science, Kista, Sweden

Abstract: *We present the use of a method – and its corresponding tool set – for compositional verification of applet interactions on an industrial smart card case study. The case study, an electronic purse, is provided by smart card producer Gemplus as a test case for formal methods for smart cards. The verification method focuses on the possible interactions between different applets, co–existing on the same card, and provides a technique to specify and detect illicit interactions between these applets. The method is compositional, thus supporting post–issuance loading of applets.*

Motivation

The growing market for smart cards and other small personal devices has increased the need to use formal validation and verification techniques in industry. These devices often contain privacy–sensitive information; this is the case in typical usages for smart cards such as health care information systems and electronic purses. Therefore strong security guarantees are needed for their wide–spread acceptance. With the acceptance of evaluation schemes such as Common Criteria[3], industry has come to realise that the only way to achieve such high guarantees is to adopt the use of formal methods in industrial practice.

Various work has been done, aiming at the verification of different kinds of properties of smart card applications. Properties under study are for example functional correctness, confidentiality, availability and restrictions on information flow.

[1]Full paper available as [3]. Research partially supported by the EU as part of the Verifi Card project IST-2000-26328.
[2]Research done while at INRIA Sophia Antipolis, partially supported by ERCIM Fellowship No. 2002-10.
[3]See `http://www.commoncriteria.com`.

Often this work focuses on the correctness of a single applet, or of a set of applets that is known in advance. However, future generations of smart cards are expected to allow post–issuance loading of applets, where newly installed applets interact with the applets already present on the card. As a consequence, at the time the card is issued, it is not known which applets it might contain. Therefore, a compositional approach is needed, where one can state minimal requirements for the applets that can be loaded later on the card, while it is verified at loading time that the applets actually respect these requirements. Only then, existing applets can safely communicate with new applets, without corrupting the security of the card.

We focus on a particular kind of properties to ensure the security of the card, namely the absence of illicit control flow between the different applets. For multi–application smart cards, certain control flow paths can be undesirable because of general platform–dependent restrictions, like the recommendation to avoid recursion due to limited resources, or due to application–specific restrictions, like undesirable information flow caused by illicit applet interactions.

Program model and verification method

We consider sequential applets without concurrent threads (as in Java Card) and model their sequential control flow as push–down automata, abstracting from all data–related aspects.

We have developed an algorithmic compositional verification technique for control flow safety properties of applets (described in detail in [5]), specified in a temporal logic language. These properties can be either *structural*, interpreting formulae over the control flow graph of an applet, or *behavioural*, interpreting formulae over applet behaviour. Our approach is compositional in that it allows global control flow properties of the whole system to be inferred from local control flow properties of the individual applets. However, while the global properties can be behavioural or structural, we require the local properties to be structural; our technique does not allow global behavioural properties to be inferred from local behavioural ones. An important asset of our method is that the verification tasks involved can be solved algorithmically: once the local applet properties and the global illicit applet interaction are specified, all verifications can be done automatically, using push–button technology.

Technically, our method is inspired by the work of Grumberg and Long [2]. It is based on the construction of maximal applets *w.r.t.* structural safety properties. An applet is maximal *w.r.t.* a property if it simulates all applets respecting this property. More concretely, suppose we want to prove that the composition of applets A and B respects a security property, formulated as behavioural safety property ϕ. We specify structural properties σ_A and σ_B for which we construct maximal applets $\theta_{I_A}(\sigma_A)$ and $\theta_{I_B}(\sigma_B)$, respectively (where I_A and I_B are the interfaces of the applets A and B). We show, using existing model checking techniques for push–down automata, that their composition respects the behavioural safety property ϕ, *i.e.* $\theta_{I_A}(\sigma_A) \uplus \theta_{I_B}(\sigma_B) \models \phi$. Since the simulation pre–order is preserved under applet composition and logical properties are preserved by simulation, from the validity of this assertion we can conclude that $A \uplus B \models \phi$ holds for any two applets A and B satisfying σ_A and σ_B, respectively. Later, when we get concrete implementations for A and B, it remains to verify, using existing finite-state model checking techniques, whether they respect σ_A and σ_B, respectively.

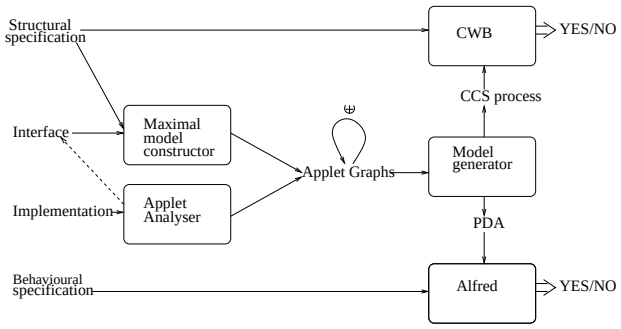


Figure 1: Overview of tool set

Tool set

To support our method, we have developed a tool set, combining existing model checkers with tools that are specific to our method. Figure 1 gives a general overview. As input we have for each applet either an implementation, or a structural property, restricting its possible implementations, plus an interface, specifying the methods provided and required by the applet. For these inputs, we construct an applet representation, which is basically a collection of control flow graphs representing methods, plus the applet interface. In case we have the applet implementation, we use the *Applet Analyser* to extract the applet graph. In case we have a structural property, we use the *Maximal Model Constructor* to construct a maximal applet graph, simulating all possible implementations of applets respecting the formula. Applets can be composed, using the *applet composition* operator $\uplus$. This operation essentially corresponds to forming the disjoint union of applet control graphs. Using the *Model Generator* the resulting applet graphs are translated into models which serve as input for different model checkers. To check structural properties, we translate the resulting graphs into CCS processes, which can be used as input for the Edinburgh Concurrency Workbench (CWB) [1]. To verify a behavioural safety property of a composed system, we translate the composed graphs into a push–down automaton, which forms the input for the model checker Alfred [4].

Case study

To evaluate the usefulness of our approach, we applied it to an electronic purse case study, provided by Gemplus, defining applications *Purse* and *Loyalty*, among others. Every time the card holder wishes to join a loyalty program, the appropriate applet can be loaded on the card. Subsequently, the purse and the different loyalties will exchange information about the purchases made, so the loyalty points can be credited.

For efficiency reasons, the electronic purse keeps a log table of all credit and debit transactions, and the loyalty applets can request the (relevant) information stored in this table. Further, loyalties might have so–called partner loyalties, which means that a user can add up the points obtained with the different loyalty programs. Therefore, each loyalty should keep track of its balance and a so–called extended

balance. If the user wishes to know how many loyalty points exactly are available, the loyalty applet will ask for the relevant entries of the purse's log table in order to update its balance, and it will also ask the balances of partner loyalties in order to compute the extended balance.

If the log table is full, existing entries will be replaced by new transactions. In order to ensure that loyalties do not miss any of the logged transactions, they can subscribe to the so–called *logFull* service. This service signals all subscribed loyalties that the log table will be overwritten soon, and that therefore they should update their balances. Typically, loyalties will have to pay for this service.

Suppose we have an electronic purse, which contains besides the electronic purse itself two partner loyalties, say L_1 and L_2. Further, suppose that L_1 has subscribed to the *logFull* service, while L_2 has not. If in reaction to the *logFull* message L_1 always calls an interface method of L_2 (say to ask for its balance), L_2 can implicitly deduce that the log table might be full. A malicious implementation of L_2 might therefore request the information stored in the log table before returning the value of its local balance to L_1. If loyalties have to pay for the *logFull* service, such control flow is unwanted, since the owner of the *Purse* applet will not want other loyalties to get this information for free.

Specification and verification

To give formal specifications for the global and local properties, we have developed a set of specification patterns, both at structural and behavioural level, which define formulae in the modal μ-calculus. In this summary, we only give an intuitive explanation of the safety properties; for their formalisation we refer to the full paper.

Global Security Property To guarantee that no loyalty will get the opportunity to circumvent subscribing to the *logFull* service, we require that if the *Purse* calls the *logFull* method of a loyalty, within this call the loyalty does not communicate with other loyalties. However, as the *logFull* method is supposed to call the *Purse* for its transactions, we also have to exclude indirect communications, via the *Purse*. We require the following global behavioural property:

> A call to *Loyalty.logFull* does not trigger any calls to any other loyalty.

Property Decomposition Next, we phrase local structural properties for *Purse* and *Loyalty*. Within *Loyalty.logFull*, the *Loyalty* applet has to call the methods *Purse.isThereTransaction* and *Purse.getTransaction*, but it should not make any other external calls (where calls to shareable interface methods of *Loyalty* are considered external[4]). Thus, a natural structural property for *Loyalty* would be, informally:

> From any entry point of *Loyalty.logFull*, the only reachable external calls are calls to *Purse.isThereTransaction* and *Purse.getTransaction*.

To specify a structural property for the *Purse* applet, we know that within a call to *Loyalty.logFull* it can only be activated via *Purse.isThereTransaction* or *Purse.getTransaction*. Therefore, we propose the following property.

[4]Notice that since we are performing class–based analysis, we cannot distinguish between calls to interface methods of other instances, and those of the same instance.

> From any entry point of *Purse.isThereTransaction* or *Purse.getTransaction*, no external call is reachable.

Using our tool set we show that: (1) the local properties are sufficient to establish the global security property, and (2) the implementations of the different applets respect the local properties.

Future work

The case study shows that the presented verification method and tool set can be used in practice for guaranteeing absence of illicit applet interactions. However, there are some possibilities for improvement. Finding suitable local properties, which requires ingenuity, is sometimes complicated by the limited expressiveness of the structural specification language. Another difficulty stems from the inherent algorithmic complexity of two of the tasks: both maximal model construction and model checking behavioural properties are problems exponential in the size of the formula, thus making heuristic optimisations of these algorithms crucial for their successful application. For many useful properties, the size of the formula depends on the size of the interface. Therefore, it is crucial to develop techniques to abstract from method names which are irrelevant to the given property.

Future work will thus go into fine–tuning the notion of interface, by defining public and private interfaces. Now interfaces contain all methods provided and required by an applet. We wish to restrict the verification of the global safety properties to public interfaces, containing only the externally visible methods, provided and required by an applet. In order to check whether an implementation respects its local property, we will need to define an appropriate notion of hiding. We also intend to extend the set of specification patterns that we use, by investigating which classes of security properties generally are used. Finally, on a more theoretical side, we will study whether we can extend the expressiveness of the logic used (*e.g.* by adding diamond modalities) and under what conditions we can allow local behavioural properties.

References

[1] R. Cleaveland, J. Parrow, and B. Steffen. A semantics based verification tool for finite state systems. In *Proc. 9th IFIP Symp. Protocol Specification, Verification and Testing*, 1989.

[2] O. Grumberg and D. Long. Model checking and modular verification. *ACM Trans. on Prog. Lang. & Syst.*, 16(3):843–871, 1994.

[3] M. Huisman, D. Gurov, C. Sprenger, and G. Chugunov. Checking absence of illicit applet interactions: a case study. In Michel Wermelinger and Tiziana Margaria, editors, *Fundamental Approaches to Software Engineering, FASE 2004*, number 2984 in LNCS, pages 84–98. Springer, 2004.

[4] D. Polanský. Verifying properties of infinite-state systems. Master's thesis, Masaryk University, Faculty of Informatics, Brno, 2000.

[5] C. Sprenger, D. Gurov, and M. Huisman. Compositional verification for secure loading of smart card applets. In *Memocode 2004*, 2004. To appear.

Minutes of the EASST General Assembly
March 30, 2004, Barcelona

Report by Tiziana Margaria* and Hartmut Ehrig **
*University of Göttingen
**Technical University Berlin

The EASST General Assembly in 2004 took place on March 30th, during ETAPS 2004 in Barcelona and it was chaired by the EASST President Tiziana Margaria. Since the General Assembly was open not only to EASST members, but also to all other ETAPS participants, the report of the president started with a short summary of the more recent developments of EASST, founded in 2000, having now about 200 members and having elected the current Board and presidential team in 2003. In this respect, the General Assembly unanimously ratified the election of the Board Members (Egidio Astesiano, Hartmut Ehrig, Marie-Claude Gaudel, Susanne Graf, Tiziana Margaria, Julia Padberg, Herbert Weber, Michel Wermelinger) and of the EASST President (Tiziana Margaria), Vice President (Hartmut Ehrig), Treasurer (Herbert Weber), and Secretary (Julia Padberg).

The president reported on the ongoing activities to establish a network of synergetic cooperation with other professional associations, like Formal Methods Europe (FME), the Society for the Design and Process of Systems (SDPS), the Software Engineering Society (SES), the ERCIM Working Group on Formal Methods for Industrial Critical Systems (FMICS). The cooperation will concern mutual exchange of information on ongoing initiatives and activities, with involvement of thematically adequate representatives of the cooperating associations, informing the members about those activities and possibly with support by means of prizes and mutual *in cooperation with* status of the events.

Michel Wermelinger, Co-Chair of FASE 2004 together with Tiziana Margaria, reported on the current FASE, underlining its very positive development both in terms of submissions, raised to 98 (91 regular papers and 7 tool submissions), and in terms of quality of the presentations and thematic spread of the sessions.

The Vice President Hartmut Ehrig has reported on the International Conferences on Graph Transformations (ICGT) in 2002 and forthcoming in 2004. ICGT 2002 was the first international conference on this topic after a successful series of international workshops

between 1978 and 1998. The first conference was excellently organized by Fernando Orejas in Barcelona and will be continued by ICGT 2004 in Rome, Sept 29 – Oct 2.

It was finally reported that the organizers of FMICS, VMCAI, and ISoLA had expressed their interest to achieve *in cooperation with* status for their next events, possibly in conjunction with EASST Best Paper Awards in form of a certificate for excellent contributions in the area of Software Science and Technology. There was unanimous expression of support by the Board and by the Assembly.

To establish and cultivate the connection to particularly active areas of Software Engineering and Software Technology, the President proposed to establish a group of Thematic Representatives, who should operate in close connection with the EASST Board and the National Representatives in order to foster the interaction of EASST with the specific communities that work in their thematic area and to promote bidirectional dissemination on a steady basis. The idea was strongly supported very and made precise operationally during the subsequent Board Meeting (see also the specific report *Guidelines for EASST Representatives* by the Vice President Hartmut Ehrig in this newsletter).

Julia Padberg presented the development of the EASST Newsletter, which reports about all the activities of EASST and features a number of columns devoted to specific topics of central interest to EASST. The following columns are now established:

- The Tool Column: edited by Tiziana Margaria
- The Continuous Learning Column: edited by Herbert Weber
- The Software Architecture Column: edited by Michel Wermelinger and Ralf Reussner
- The Visual Modeling Techniques Column: edited by Reiko Heckel
- The Software Analysis and Optimisation Column: edited by Jens Knoop

These will be soon flanked by newly established ones under the editorial responsibility of the Topic Representatives.

In 2003-2004 the following contributions were awarded EASST prizes for the best paper in the area of Software Science and Technology:

- at ETAPS 2003, Warsaw, the FASE paper:
  Spatial Security Policies for Mobile Agents in a Sentient Computing Environment, by D.J Scott, A. Mycroft, A.R.Beresford,

- at FMICS 2003, the 8th Workshop on Formal Methods for Industrial Critical Systems held in Røros (Norway) in June 2003:

- Paul Ziemann, Martin Gogolla (University of Bremen): *Validating OCL Specifications with the USE Tool - An Example based on the BART Case Study*
- Stefan Blom, Simona Orzan (CWI): *Distributed State Space Minimization.*

A summary version of these contributions appeared in the EASST Newsletters Vol. No. 7, December 2003,

- at ETAPS 2004, Barcelona, the FASE paper
  Checking Absence of Illicit Applet Interactions: A Case Study
  by Marieke Huisman, Dilian Gurov, Christoph Sprenger, and Gennady Chugunov (a summary version appears in this Volume of the EASST-Newsletter).

Additionally, Fernando Orejas raised the problem of the recent need of US-based, but worldwide active, professional organizations to respond to a governmental reminder of embargos wrt. some countries. This has led to different reactions from those organizations, which ranged from ignoring the reminder to restricting the services to members from those countries. The Assembly openly discussed the issue, concluding with the decision that the EASST statute should explicitly exclude discriminations of any kind to its members on a political basis, and delegated to the Board the concrete issue of evaluating whether EASST should take an official position on this matter.

The next EASST General Assembly will take place during ETAPS 2005 in Edinburgh, Scotland.

Guidelines for EASST Representatives

Hartmut Ehrig (EASST Vice President)
Technical University Berlin

The EASST Board has agreed on the following guidelines for regional and topic representatives for EASST:

1. General Guidelines

The EASST Board appoints national and topic representatives of EASST for the election period of the board. After a new election of the board the appointments can be continued several times. Each representative is requested to report about her/his activities at least once a year in the EASST Newsletter and is invited to take part in the EASST Board Meetings.

2. Regional representatives

The regional representatives of EASST are responsible for the EASST activities in specific countries or regions inside or outside of Europe. Although the main activities of EASST are within Europe, EASST is also interested in membership and cooperations outside Europe. The EASST representative is required to get into contact with important software organisations and companies in her/his country or region and to establish links or even sponsorships of EASST for national SE-conferences and vice versa sponsorships of SE-companies for EASST.

Current regional representatives

Fernando Orejas: Spain
Jose Mesenguer: USA
Carlo Montangero: Italy
Andrzej Tarlecki: Poland
Heinrich Hussmann: Germany
Susanne Graf: France
Mohammed Bettaz: Middle East and North Africa

3. Topic Representatives

The topic representatives of EASST are responsible for a specific topic within the scope of EASST. Especially - by invitation of the EASST Board - they may become editors of corresponding column for this topic in the EASST Newsletter, where they can invite other authors for contributions concerning international activities related to this topic. The EASST representative is especially responsible for getting into contact with the organisers of international conferences and workshops in her/his topic area and to make proposals for sponsorship of EASST, which are decided by the EASST Board. For important conferences there can be an agreement between the organiser and the EASST Board to offer an EASST Award for the best software engineering paper.

Current Topic Representatives for EASST and Column Editors

The Tool Column: Tiziana Margaria
The Continuous Learning Column: Herbert Weber
The Software Architecture Column: Michel Wermelinger and Ralf Reussner
The Visual Modeling Techniques Column: Reiko Heck el
The Software Analysis and Optimisation Column: Jens Knoop

The Working Group ”Software Architecture” of the German Computer Science Society (GI-AK SoftArch)

Ralf H. Reussner *
*University of Oldenburg / OFFIS, Germany

Abstract. *We report on the foundation of the new working group ”Software Architecture” of the German computer science society (GI-AK SoftArch). The working group aims to support the practical application and research of software architectures. This support is given (a) by establishing a communication platform and (b) by a joint book project for a German handbook on software architecture. Specifically, the working group tries to focus their efforts by organising and co-ordinating contributions of chapters for this handbook. Besides the organisation of the handbook, keynotes on the role of software architecture and its current state (from a research and a practical viewpoint) and its modelling with the novel UML2 were given. As a part of the German Computer Society (G.I.) the working group has a national focus. However, international participation is welcomed. The foundational meeting took place on February, 11th 2004 at the OFFIS research institute on Oldenburg and attracted 23 participants all across Germany. Organisationally, the AK SoftArch belongs to the OOSE (object oriented software engineering) group of the G.I.*

Keywords: WG Software Architecture

1 Motivation

The foundation of a GI working group dedicated to software architecture is motivated by the increasing importance of software architectures. In practice, architectural modelling is not used as a means by itself but architectural modelling and analyses can be seen as a pre-requisite or at least a benefit to many areas of today’s software engineering. For example in legacy system integration, it is hard to plan a migration if the target architecture and its components are not made explicit. Likewise, the establishment of a product line (regardless whether it is planned top-down by domain models or, as most often the case, is developed bottom-up from existing products) needs explicit architectural models. Besides these ”applications” of architectures, in general architectures serve as a means for communication among the different developers and the various stakeholders. From a researcher’s perspective, the promise of software architectures to systematically deal with non-functional properties (including externally visible quality attributes) is still a challenging field of research. However, as systematically dealing with quality

attributes is a characteristic of any engineering discipline, the use of software architectures and associated analysis techniques plays an important role toward making software engineering a true engineering discipline.

Opposed to this importance of software architectures from a practical and a research perspective, Germany lacks a communication platform for exchanging knowledge and experience on software architectures. The GI working group on software architectures is thought to provide such a platform. To focus this ambition, the working group has the concrete goal to publish a German handbook on software architecture. This handbook – and the working group as a whole– intents to counteract the gap between industrial software development practice and academic software engineering research by supporting the transition of software engineering knowledge on architectures from research into practice and to increase its practical benefits by research.

The idea of a working group on components and architectures was originally discussed by Sven Overhage, Ralf Reussner and Steffen Becker during ECOOP in June 2003 in Darmstadt. In September 2004, a first proposal worked out by Willi Hasselbring and Ralf Reussner was presented by Ralf at the meeting of the subdivision (”Fachgruppe”) ”object oriented software engineering (OOSE)” during the NetObjectDays in September in Erfurt.

As detailed protocol of the foundational meeting on Februaray 11, written by Steffen Becker, is available in German under
`http://se.informatik.uni-oldenburg.de/GIAKSoftArch/Treffen11022004`. In this report, we summarise central points.

2 Summary of the Foundational Meeting

After the presentation of the above motivation of the working group, the programme continued with three keynotes. Firstly, Ralf Reussner presented from a researcher’s perspective the state of the art of software architectures and its role for an engineering approach to software construction. In particular, he emphasised that from a research perspective, the mere syntax (or graphical notation) for specifying components is of lower interest compared to approaches to *make use of* architectural models, i.e., actual algorithms for analysing architectures. This corresponds to the view, that a mere specification cannot be a means for itself, but must provide additional benefit to justify the costs of specification. This justification, for example, can be given by analysis tool evaluating the architecture according to certain properties, such as quality attributes (e.g., reliability or performance) of other non-functional properties (e.g., maintainability, re-usability, etc).

The second keynote on software architectures by Johannes Siedersleben gave a very experienced overview on the state of the art from a practitioner’s perspective. His talk emphasised the importance of architectures as a communication platform for the various stakeholders of a project. Therefore, architecture descriptions need to include component and interface specifications. This includes specifications of exceptions and error handling on an architectural level which is still an open issue for research. Standardizing error handling is just a special case of a broader topic: The separation of applications specific logic and code dealing with technical issues. As an outlook, architectures for federated systems were discussed.

Last but not least, Mario Jeckle presented a lively talk on using the UML2 for architectural modelling. After a brief presentation of the history of UML and an overview, the novel diagrams of UML2 and changes compared to UML1.x were discussed. Then, novel diagramme types for architectural modelling were presented in detail, including the composition structure diagram, the component diagram, the communication diagram and the deployment diagram. The talk concluded with general architectural principles for modelling architectures with the UML2.

Peter Tabeling distributed a statement on the importance of architecture as a communication platform and asked for the establishment of the topic ”architectures as a means for communication during the software development process”. The participants totally agreed in this important role of architectures and accepted his proposal. The text can be found under `http://se.informatik.uni-oldenburg.de/GIAKSoftArch/Treffen11022004/InitiativeModelleAlsKommunikationsmittel.pdf`. According to this document, the reason why software architecture does not play that in role for communication in many current projects is grounded by deficiencies of notations, terminology and processes.

The participants agreed on focussing on the following *topics of interest*:

- Modelling and specifying architectures, architectures as a means for communication, guidelines for modelling, glossary
- Interoperability and adaptation (connectors and adaptors) of components
- COTS and legacy system integration, migration of architectures
- Analysis and evaluation of architectures
- Prediction of quality attributes
- Reuse of architectures (including patterns, styles, frameworks, product lines and families)
- Implementation of architectures
- Robustness of architectures and error handling on an architectural level
- Processes for architecture creation and management
- Architectures as a means for communication, project planing and cost estimation
- Education, teaching and training of software architecture knowledge, the professional role of an architect.

Wilhelm Hasselbring proposed Ralf Reussner as a speaker of the working group, who then was elected unanimously. The working group is lead by an organisational committee (”Leitungsgremium”), whose members are: Steffen Becker, Simon Gieseke, Willi Hasselbring, Stefan Krieghoff, Sascha Müller, Sven Overhage, Ralf Reussner, Ulf Schreier, Wolfgang Strunk, Peter Tabeling. Organisationally, the working group on software architecture is a part of OOSE of the GI.

3 The Handbook on Software Architecture

The aim of the handbook is to support the use of software architectures in practice. Therefore, the primary goal is to give an overview on existing notations, technology and methods associated to software architecture. Consequently, the description of original and novel ideas of a specific research project are of lower concern. Much more, we are interested in thought-out validated approaches and experience reports.

During discussing a prepared organisation (table of content) of the handbook, we realised that the actual organisation will to a certain extent also emerge bottom-up depending from the actual chapter submissions. However, the tight interaction between authors, enabled by the organisation of the working group, should be used to co-ordinate the submissions in a closer manner as it is common for normal conference proceedings or article collections.

Currently, the book roughly is structured as follows:
Part I: Description of Architectures
Part II: Use of Software Architectures
Part III: Example Architectures and Their Deployment
Part IV: Outlook and Prospects (including the education and occupational image of software architects.)

Tool descriptions are thought either as an appendix (as their timeliness vanishes fast) or as a recurring topic throughout the chapters.

In particular, it was pointed out that each chapter should have a list of 10 best practices regarding the deployment of the described concepts.

4 Future

The next meeting will be helt in Potsdam at the Hasso-Plattner-Institute for Software Systems Engineering (HPI) on June, 24th and 25th. The primary goal of this meeting is the presentation of book chapter proposals. Derived from these presentations and the written proposals, the chapters are grouped according to their content and the chapter structure of the book will be worked out. Secondarily, the meeting will serve as platform for discussing software architectural topics in general and for exchanging experience and results.

The overwhelming interest in this working group, expressed by the broad participation of the foundational meeting and the steadily increasing number of mailing list subscriptions (currently above 70) shows the need of such a communication platform for software architecture. In case of interest, please subscribe by sending an email with the command `subscribe aksoftarch` in the subject line to our mailing list `majordomo@listserv.uni-oldenburg.de` or visit our web-page at

`http://se.informatik.uni-oldenburg.de/GIAKSoftArch/`

List of Participants of the Foundational Meeting on February, 11th, 2004 at the OFFIS in Oldenburg

Steffen Becker	Uni Oldenburg
Gerd Beneken	TU München
Ludger Bischofs	Uni Oldenburg
Susanne Boll	Uni Oldenburg
Viktoria Firus	Uni Oldenburg
Simon Giesecke	RWTH Aachen
Jörg Graf	PARCS
Bernhard Gröne	HPI
Willi Hasselbring	Uni Oldenburg
Mario Jeckle	FH Furtwangen & DaimlerChrysler Research
Stefan Krieghoff	KDO
Jasminka Matevska-Meyer	Uni Oldenburg
Jürgen Meister	OFFIS
Sascha Müller	Uni Erlangen
Jan Ploski	OFFIS
Ralf Reussner	Uni Oldenburg
Ansgar Scherp	OFFIS
Ulf Schreier	FH Furtwangen
Johannes Siedersleben	FH Rosenheim & sd&m Research
Wolfgang Strunk	ST-Werkstatt
Peter Tabeling	HPI
Thorsten Teschke	OFFIS
Thomas Wendt	Uni Leipzig

Language Engineering for Model-driven Software Development

Reiko Heckel *and Jean Bézivin **
*Universität Dortmund, Germany
**University of Nantes, France

Abstract. *This paper summarizes the objectives and structure of a seminar with the same title, held from February 29th to April 5th 2004 at Schloss Dagstuhl, Germany.*

Keywords: MDA, language design, technological spaces

1 Introduction

Model-driven approaches to software development require precise definitions and tool support for modeling languages, their syntax and semantics, their notions of consistency and refinement, as well as their mappings to the implementation level. In order to support model-driven development in a variety of contexts, we must find efficient ways of designing languages, accepting that definitions are evolving and that tools need to be delivered in a timely fashion.

In this respect, language definitions are not unlike software. Thus, a discipline of *language engineering* is required to support the design, implementation, and validation of modeling languages with the goal to deliver languages at low cost and with high quality.

An important contribution of any engineering science, besides the actual technology provided, is the meta knowledge about what are the relevant concerns to be addressed, what are the possible solutions, and what concern is best addressed in a given context by which kind of technology.

It is understood that different concerns of language engineering, like the definition of abstract syntax and well-formedness rules, operational and denotational semantics, consistency and refinement relations, and model transformations, will, in general, require technologies from different domains.

A framework for classifying, choosing, and relating different solutions domains is provided by the concept of *technological spaces* [KBA03]. A technological space is a working context with a set of associated concepts, body of knowledge, tools, acquired skills and possibilities, often associated to a given community. Well-known examples include XML, UML meta modeling, graph transformation, algebra and logic, programming languages, etc.

It has been the goal of the seminar to investigate relevant concerns and promising solution domains for language engineering, learn from specific solutions presented by the participants, and attempt a pro-

visional classification and mapping. To illustrate problems and available solutions, a sample language engineering problem was proposed and elaborated.

After a more detailed discussion of the architectural aspect of language engineering, this summary presents this case study, discusses concerns and open issues raised by the corresponding language definition problem, and gives a more general motivation of technological spaces as solution domains for model-driven development.

2 MDA as Architecture of Processes and Languages

With its Model-Driven Architecture (MDA) initiative, the OMG has placed models into the center of their vision of software development. The claim is that, using UML models, "platform-independent application descriptions ... can be realized using any major open or proprietary platform, including CORBA, Java, .NET, XMI/XML, and Web-based platforms" [Gro04]. Technically, the approach is based on model transformations for the automated refinement of platform-independent by platform-specific models, and code generation from the latter to the actual implementations.

As means of application integration, a model in MDA has a primarily architectural purpose (hence the name), even when it does not describe the architecture of a system. Consider, as a simple but typical example, a UML class diagram providing a platform-independent data model for an application structured according to a three-tier architecture into presentation layer, application logic, and data base access. To ensure the consistency of the data models used in the different layers, they shall all be derived from a single integrated model, i.e., the platform-independent class diagram.

In the course of development, the relevant parts of this diagram could be transformed into platform-specific models for, e.g., an XML schema, and a Java class structure, and a relational data base schema, in order to provide adequate representations for the data in different layers. From these platform-specific class diagrams the actual schemas or classes could be generated automatically.

In this case, the architectural aspect is not present in the models themselves, but reflected by the transformations of models and the consistency relations between them. In this sense, we think of the "A" in "MDA" as referring to the *architecture of the development process*.

As a consequence of the MDA goal of generating implementations from models automatically, the quality of these models (their level of precision and formality, their completeness and consistency) must be comparable with actual programs or formal specifications. However, a model can only be as good as the definition of the language to which it conforms. Thus, the first challenge consists in delivering high quality language definitions. Due to the necessary specialization of languages to different implementation platforms and application domains, this challenge manifests itself for a variety languages and dialects which, moreover, are continuously evolving. Hence, *reuse* is inevitable to reduce costs in defining and implementing languages. That means, languages (or their respective definitions) should be seen as "components" with "connectors" defined by consistency relations and transformations defined at the language level. This leads us to a view of MDA as an *architecture of languages*.

Continuing the architectural metaphor, a model used in a concrete development project, conforming to its language definition, can be seen in analogy to a concrete component (instance) in the configuration of a system, instantiating a generic component (type).

In the following section, we present an example of the first perspective, the architecture of the development process, by means of a sample approach to the model-based development of Web service processes. Section 4 is devoted to the second view of MDA as an architecture of languages.

3 A Sample Language Engineering Problem

We sketch a model-based approach to the development of Web service processes and discuss the concerns and issues raised by its definition and tool support.

3.1 Approach

A *Web service* is a software component that can be dynamically discovered, linked, and invoked by its clients via XML-based protocols. This software-oriented definition of the term can be contrasted with a business-oriented view, considering a Web service as a *business process*, implemented by the composition (and coordination) of simpler services provided by other businesses.

The composition of services provided by different independent parties, at both development time or runtime, requires a high degree of standardization and flexibility. Therefore, rather than hard-coding business processes in platform-specific programming languages which depend on certain compilers and run-time environments, platform-independent XML-based languages like the *Business Process Execution Language for Web Services (BPEL4WS)* [ACD+03] are advocated. Such processes in XML representation can, at least in theory, be adapted at runtime, exchanged between different services, and executed on different standardized interpreters.

To support the development of BPEL processes in a model-based approach, we require

1. an intuitive and adequate *modelling notation* to allow precise specifications of processes at the conceptual level;
2. an *automatic transformation of process models to their XML-based encoding* to avoid the costly and error-prone task of deriving the implementation manually;
3. *techniques to analyze processes at the model level for syntactic and semantic properties* to avoid "debugging" the XML code.

These problems and requirements are prototypical for a wide variety of languages and platforms in the Web services domain and elsewhere. Therefore, instead of defining and implementing languages, transformations, and analysis tools for every single problem, reusable solutions are required.

In [HV04] we have presented an approach based on the combination of three such solutions: the *Unified Modeling Language (UML)* [Obj03] as standard notation for modelling software, *graph transformation* [Roz97] as meta language for defining model transformations, and a semantic interpretation of process models in terms of *Communicating Sequential Processes (CSP)* [Hoa78] which offers a language to express semantic consistency properties and tool support for analysis.

An outline of the approach is given by the diagram in Fig. 1, whose vertices are the languages by which processes may be represented, and whose edges represent uni- or bi-directional transformations between these representations.

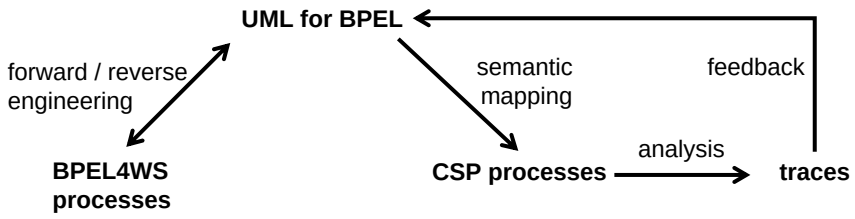


Figure 1: Outline of the approach: languages and transformations

3.2 Concerns and Open Issues

The example raises several concerns that are prototypical to model-driven approaches.

Syntactic and semantic extensions. The UML, as a general-purpose modelling language, provides a rich set of concepts to model all kinds a software system. However, to address the more specific concerns of a particular application domain or implementation platform the language needs to be specialized and extended. For this purpose, the standard [Obj03] foresees the extension mechanism of *profiles*, a compromise between desirable flexibility of the language and necessary compatibility with existing tools. We have to use profiles to tailor, in particular, the syntax of UML activity diagrams to the specification of BPEL4WS processes. However, this tailoring should not stop at the level of syntax, but continue to provide semantics to the new and specialized language constructs. To define semantics in an incremental, extensible way represents an problem that is not even completely solved for classical programming languages, but yet more relevant for the UML extension mechanism.

Model Transformations. In our example, model transformations occur in several places: the transformation of activity diagrams into BPEL4WS, the implementation language and into CSP, the language for behavioral analysis, as well as the transformation of traces, the result of CSP model checking, back into models. In many situations, two-way transformations are required, e.g., to support a *round-trip engineering* approach, where not only models are transformed into implementations (forward engineering), but also vice versa (reverse engineering), thus allowing incremental changes at both levels.

Moreover, for a transformation specification to be manageable and reusable, a modular approach is important which is structured in terms of the fundamental concepts of the domain. In this case, whenever a concept is added or modified, the corresponding transformation rules can be exchanged, hopefully without affecting the rest of the mapping specification. For example, in the domain of executable business processes, or workflow models, a corresponding concept analysis has produced an established list of *workflow patterns* [vdAtHKB03], a subset of which is supported by UML activity diagrams. In [HV04], we have given a mapping specification based on graph grammars which uses workflow patterns to organize the set of rules. However, this mapping is restricted to well-structured (essentially hierarchic) activity diagrams. It is open if a similar modularization can be supported in general.

Model-based Analysis. The final building block of our approach is the analysis of processes. Depending on the representation on which the analysis is performed, we distinguish between *syntactic* and *semantic* analysis. The former is often restricted to the evaluation of well-formedness constraints on (the abstract syntax of) the model which reveals inconsistencies in structural dependencies and typing.

Analysis of behavioral properties, instead, can hardly be done at the syntactic level, but requires a mapping of models into a semantic domain providing (1) a representation of the behavior to be analyzed, (2) means to express the desired properties, and (3) techniques and tools to check if these properties hold [EKGH01]. We have chosen the semantic domain of CSP [Hoa78] for this purpose, whose refinement relations are the basis for expressing properties over processes while tool support is provided by the FDR2 model checker [Ros97].

However, both syntactic and semantic analysis, should they be automated, are limited to those parts of the model that are completely formalized. Inscriptions in natural language, for example, can only be checked manually by a review process. Still, semi-formal models have their advantages over formal ones if they are used primarily by humans. The application of formal analysis techniques to incomplete and semi-formal models is an open problem.

Summarizing, the model-driven development problem for Web service processes as discussed in items 1 – 3 above could be realized through the technological spaces supporting UML models, XML documents, and CSP processes and analysis. The mapping of MDA problems to existing solution domains is discussed more generally in the next section.

4 Mapping Problems to Solutions

In order to understand such mappings for the emerging field of model-driven software development, the main characteristics of both the problem and the solution domain need to be understood.

On the problem side, one of the main goals is the separation of business-oriented and platform-related parts. This shall allow to deal more easily with rapidly changing platforms, using generative techniques to map business-neutral descriptions onto specific execution platforms. But mappings may also apply in the reverse direction, extracting business models from legacy systems.

This description covers only part of the problem, focussing on a single aspects of the development process. The separation and subsequent integration of business and platform is part of the more general task of aspect separation and aspect weaving. For example, the separate expression and merging of functional and non-functional aspects are essential parts of the general problem space, too. For each aspect we need a domain specific language to express it in a user-oriented and non-ambiguous way, as well as a corresponding integration strategy.

On the solution side, there are different alternatives, too. For example, aspect separation and weaving may be done on a code-centric basis which leads to an AOP-like paradigm. Instead, it is also possible to use model-based techniques to handle the various aspects. Since a model captures only a specific view of the overall system, it is very natural to build a dedicated model representing a given aspect.

An understanding of this relation of *representation* between a system and its model is the basis of model-driven development. But the model itself is written in a given language which, in the MDA space, is defined by a metamodel. The relation between a model and its metamodel is a *conformance* relation, the precise specification of which is the task of the language designer. Similar organizations can be found in various technological spaces.

Systems are becoming more complex because of the increase in complexity (of code, data and aspects), in evolutivity and heterogeneity. In order to build these systems, many technologies are offered. Usually

one will need to use several of these to solve a given problem. Unfortunately there is not and there will not be any uniformly superior technology because each one has its weak and strong points. In order to propose an agile method to problem solving in the domain of software system development and maintenance, the main characteristics of various technological spaces need to be identified and compared.

Some technological spaces offer better support for separation of concerns, for executability, for transformations, for modularity, etc. Among the classical spaces, we may mention model-driven engineering, XML document management, ontology engineering, programming languages and abstract syntaxes, graph theory and graph transformation, relational data base management systems, etc.

Each technological space is based on a central representation system (text files, trees, graphs, hypergraphs, etc.). This representation system may be explicitly defined, e.g., by a representation ontology or by a meta-metamodel (like the OMG MOF). This often gives rise to a three-level organization. In MDA the levels are called M3 for the MOF meta-metamodel, M2 for the metamodels, and M1 for the models. In programming languages, a similar organization holds with, e.g., EBNF at level M3, the syntax definitions of specific languages at level M2, and programs at level M1. In ontology engineering we have a corresponding structure with meta-level, intention, and extension ontologies. And in the XML technological space, this same layering could be exhibited again. We thus see that many technological spaces are similarly organized. This facilitates the construction of bridges between them. For example a brigde between MDA and XML is called XMI; a bridge between MDA and Java is called JMI; etc.

A Java program may also be considered as an XML document based on a given DTD (JavaML for example) or as an MDA model based on a Java metamodel, etc. These three representations in different technological spaces have each their advantages and drawbacks. When solving a given problem, the engineer should have a flexible and agile attitude towards which technology or mix of technologies to apply in order to solve this problem. The sample language engineering problem described in the previous section, for example, could be solved in specific instances and combinations of the programming language space, the XML document space, and the MDA space.

The study of language engineering problems from the perspective of various technological spaces is of conceptual and practical interest. At the heart of this study lies the investigation of how each space implements the basic structure required (the two relations of representation and conformance) as well as the investigation of mappings between spaces preserving that structure.

5 The Seminar

The audience of the seminar consisted of about 40 researchers and practitioners. They were representative of the different technologies which could contribute to a discipline of language engineering, i.e. UML and Metamodelling, Graph Transformation and Graph Grammars, CASE Tools, Aspect-oriented Software Development, and Programming Language Semantics.

In order to address the classical language engineering issues, discussion groups on Syntax and Semantics, Model Transformation and Consistency, Tools, and Pragmatics have been formed.

Reports on the results of these discussions, abstracts of presentations, as well as work-in-progress papers describing new ideas developed at the seminar shall be published in the Dagstuhl proceedings at `http://www.dagstuhl.de/04101/`.

References

[ACD+03] T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. Business Process Execution Language for Web Services, Version 1.1, May 2003. http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/.

[EKGH01] G. Engels, J.M. Küster, L. Groenewegen, and R. Heckel. A methodology for specifying and analyzing consistency of object-oriented behavioral models. In V. Gruhn, editor, *Proc. European Software Engineering Conference (ESEC/FSE 01), Vienna, Austria*, volume 1301 of *LNCS*, pages 327–343. Springer Verlag, 2001.

[Gro04] Object Management Group. The architecture of choice for a changing world, MDA executive overview. http://www.omg.org/mda/executive_overview.htm, 2004.

[Hoa78] C. Hoare. Communicating sequential processes. *Communicat. Associat. Comput. Mach.*, 21(8):666–677, 1978.

[HV04] R. Heckel and H. Voigt. Model-based development of executable business processes for web services. In W. Reisig and G. Rozenberg, editors, *Proc. Advanced Course on Petri Nets, Eichstätt, Germany*, LNCS. Springer-Verlag, 2004. to appear.

[KBA03] I. Kurtev, J. Bézivin, and M Aksit. Technological spaces: an initial appraisal. http://www.sciences.univ-nantes.fr/lina/atl/publications/PositionPape 2003.

[Obj03] Object Management Group. Unified modelling language(UML) 2.0, 2003. http://www.omg.org/uml.

[Ros97] A.W. Roscoe. *The Theory and Practice of Concurrency*. Prentice-Hall, 1997.

[Roz97] G. Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation, Volume 1: Foundations*. World Scientific, 1997.

[vdAtHKB03] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. *Distributed and Parallel Databases*. 2003.

Introducing SegraVis: A European Research Training Network on *Syntactic and Semantic Integration of Visual Modeling Techniques*

Reiko Heckel *
*Universität Dortmund, Germany
(on leave from Paderborn)

Abstract. *This first installment of the column on Visual Modeling Techniques will present the objectives, partners, and activities of the SegraVis network, introduce four of the partner sites in more detail, and report on the School on* Foundations of Visual Modeling Techniques *recently organized by the network.*

Keywords: visual modeling techniques, research training network

1 Why we are here

SegraVis is a European Research Training Network under the Fifth Framework, running from October 2002 until September 2006. Besides a collaborative research project in the area of visual modeling, the network implements a structured training programme for researchers in this field, especially in the early stages of their career. With this aim, the SegraVis network offers two kinds of training activities.

Research-training grants: In cooperation with leading researchers in one of the network sites, post-doctoral researchers and doctoral students advance the state of the art in visual modeling techniques, thus acquiring knowledge and skills for their own carrier.

Network activities: At schools, tutorials, and workshops organized by the network, participants broaden their knowledge, present their own ideas, and get in touch with the experts.

This first edition of the column has the purpose of introducing the network, its partners, objectives, and activities. Moreover, we we will give a more detailed report on the School on *Foundations of Visual Modeling Techniques*, held in May 2004 at Schloss Dagstuhl as the major training event of the network, and take a closer look at the scientific specialties of four of the 12 network sites.

*Enquiries for open positions can be sent **at all times** to the local coordinators or the network manager, as listed in the following section.*

Permanent Call for Applications

- visit the leading experts in the field
- join us for a grant in one of 12 attractive locations throughout Europe
- for information on vacancies contact reiko@upb.de and check at www.segravis.org

2 Who we are

The 12 network members from five European countries are listed below together with the key researchers.

- *Universität Paderborn (Germany)*, the principal contractor, with Gregor Engels (network coordinator), Wilhelm Schäfer, and Reiko Heckel (network manager)
- *Universitaire Instelling Antwerpen (Belgium)* with Dirk Janssens (local coordinator) and Serge Demeyer
- *Universitat Politécnica de Catalunya, Barcelona (Spain)* with Fernando Orejas (local coordinator), Jordi Cortadella, and Gabriel Valiente
- *Technical University of Berlin (Germany)* with Hartmut Ehrig (local coordinator), Herbert Weber, and Gabi Taentzer
- *University of Bremen (Germany)* with Hans-Jörg Kreowski (local coordinator) and Martin Gogolla
- *University of Kent at Canterbury (United Kingdom)* with Peter Rodgers (local coordinator)
- *Leiden University (The Netherlands)* with Grzegorz Rozenberg (local coordinator) and Joost Kok

- *University College London (United Kingdom)* with Wolfgang Emmerich (local coordinator) and Anthony Finkelstein
- *Universitá degli Studi di Milano, Bicocca (Italy)* with Mauro Pezzé (local coordinator), Carlo Ghezzi and Luciano Baresi (both at Politecnico Milano)
- *Technical University of Darmstadt (Germany)* with Andy Schürr (local coordinator)
- *Universitá di Pisa (Italy)* with Ugo Montanari (local coordinator) and Andrea Corradini
- *Universitá di Roma, La Sapienza (Italy)* with Francesco Parisi-Presicce (local coordinator), Paolo Bottoni, and Stefano Levialdi

The most important people are, however, those for whom the network has been set up: the young researchers with a grant at one of the SegraVis sites. If you are interested who they are and what is their personal experience, go to `www.segravis.org` and follow the menu to *grants / experience*.

3 What we are doing

In this section, we motivate and describe the research topic, objectives, and approach of the network, give an overview of the main scientific events and a more detailed report on the SegraVis school.

Topic

Both in software engineering and in the more classical engineering disciplines, the use of visual notations, e.g., for documentation of requirements or communication with customers, has a long tradition. Driven by the increasing complexity of the problems such notations have become more elaborate and evolved towards *visual modeling techniques* with their own methodology and tool support.

The reasons for the success of visual modeling techniques are manyfold. Visual representations are direct and intuitive, simplifying the communication between developers and their customers. They are extremely effective at delivering useful abstractions of industrial-scale systems. This is evidenced by the success of the UML and its paradigm of model-driven development. The *Unified Modeling Language* with its many sub-languages and dialects is, indeed, one of the two main foci for the research in this project, as well as *graph- and net-based modeling techniques* like graph transformation, high-level or timed Petri nets, flow diagrams, or domain-specific net-like notations.

Model-driven approaches to software development, where models are *the* central artifacts, require precise definitions for modeling languages, their syntax and semantics, their notions of consistency and refinement, as well as their mappings into implementations. However, languages are subject to continuous evolution, and different constellations and dialects are required in different application domains, organizations, and even individual projects. In order to support model-driven development in a variety of contexts, we must find efficient ways of designing languages and processes, accepting that definitions are subject to change and extension and that tools need to be delivered in a timely fashion.

Rather than ad-hoc solutions, a discipline of *language engineering* is required to support the definition and implementation of modeling languages with respect to their abstract syntax and well-formedness

rules, operational and denotational semantics, consistency and refinement relations, and model transformations.

It is the aim of the project to develop this engineering approach, to demonstrate its applicability to a range of modeling languages, including the UML as well as graph- and net-based approaches, to employ and improve visual modeling techniques in specific application domains, and to influence the direction of global industry standards in this area.

With this motivation, the network has defined the following research goals.

Objectives

According to their focus and level of generality, we distinguish between *meta-level objectives (M)*, *language-specific objectives (L)*, and *domain-specific objectives (D)*.

M. Meta-level objectives. The primary objective of the project is to develop *meta-level solutions* for the definition, integration, and implementation of visual modeling techniques, providing support for

M1. syntax and well-formedness;

M2. static and dynamic semantics;

M3. analysis and verification;

M4. modularity, refinement, and transformation.

Orthogonally, but at the same level of generality, solutions are sought for

M5. integration of visual modeling techniques both at the syntactic and at the semantic level;

M6. integrated meta CASE tool support.

L. Language-specific objectives. Two classes of visual modeling techniques, the UML family of languages and graph- and net-based modeling languages, shall serve as practical test cases for the meta-level solutions.

L1. Engineering the UML family includes

- the definition of syntax and semantics of a UML profiles to test the approach, as well as the evaluation of these definitions for usability and tool support;
- the comparison and possible alignment of M1-M6 with OO meta-modeling approaches currently practiced in industry, including the meta-object facility MOF and XML-based approaches.

L2. Integration and classification of graph- and net-based techniques aims at reducing the gap between formally well-founded modeling techniques based on graphs and nets and approaches used in practice. This includes

- the definition and implementation of mappings between domain-specific notations used by application experts and general-purpose formalisms based on graphs and nets;
- a systematic specification and classification of graph and net-based approaches which makes explicit the semantic choices and properties to guide the potential user;
- an integration of domain-specific graph- and net-based languages with the UML, e.g., by means of profiles or transformations, which allows modelers to use both techniques within the same development process.

D. Domain-specific objectives

The third objective of the network is to employ and improve visual modeling techniques in specific application domains, including (but not limited to)

D1. modeling support for software evolution and refactoring;

D2. modeling of component-based software architectures;

D3. specification of applications with mobile soft- and hardware.

The means for achieving these objectives are discussed next.

Approach

In text-based programming and specification languages, abstract syntax trees or algebraic terms are the core models for most language-related tasks. In visual modeling techniques it is often convenient to represent the abstract syntax of diagrams by *abstract syntax graphs*, either instances of a meta model as in the case of the UML, or generated by a graph grammar. Hence, for dealing with syntax and semantics definitions, analysis and transformation of models, the primary approach is the generalization of classical, tree-based solutions to graphical structures.

The basic technology for this generalization is provided by the concepts, theory, and tools of graph transformation, a rule-based approach in the tradition of Chomsky grammars, attribute grammars, and term rewriting, i.e., those rewriting techniques that provide the foundation of textual language technology and compiler construction.

For a more detailed overview (with references) of the individual approaches to the meta level objectives, the reader is referred to `www.segravis.org` under *resources / deliverables*.

Activities

The network regularly organizes and supports a number of workshops and conferences:

- The *International Conference on Graph Transformation (ICGT)* is held bi-annually in September or October. Its first installment took place in 2002 in Barcelona (Spain) October 2002. The second

ICGT will be in Rome (Italy) September 28 - October 2, 2004 in collocation with the *Conference on Visual Languages and Human-Centric Computing (VL/HCC)*.

Satellite events of relevance to the network's objectives include workshops on

- Visual Languages and Formal Methods (with VL/HCC);
- Petri Nets and Graph Transformations;
- Term Graph Rewriting;
- Graph-Based Tools;
- Software Evolution through Transformations;

as well as tutorials on

- Foundations and Applications of Graph Transformation;
- DNA Computing and Graph Transformation.

While the deadlines of the main conferences ICGT and VL/HCC have passed, ***satellite workshops are still accepting submissions****, check the respective Web pages at* `www.dsi.uniroma1.it/icgt2004/` *and* `HCC-URL`.

- The *Workshop on Graph Transformation and Visual Modeling Techniques (GT-VMT)* is usually associated with major conferences, e.g., 2002 with ICGT and 2004 with ETAPS, both in Barcelona.
- The *Workshop on Application of Graph Transformation with Industrial Relevance (AGTIVE)* has been held 1999 in Kerkrade (The Netherlands) and 2003 in Charlottesville, Virginia (USA).
- The *Workshop on Uniform Approaches to Graphical Specification Techniques (UniGra)* is held bi-annually with ETAPS, i.e., 2001 in Genova (Italy) and 2003 in Warshaw (Poland).
- The *Symposium on Visual Languages and Formal Methods (VLFM)* has been held as part of the IEEE Conference on Human-Centric Computing (HCC), 2001 in Stresa (Italy) and 2003 in Auckland (New Zealand), and is continued as a satellite workshop of VL/HCC 2004 in Rome.

As major training event, the network organized the School on *Foundations of Visual Modeling Techniques*, held in May 2004 at Schloss Dagstuhl.

School

The School has been held in the week of May 3 – 7 at the International Conference and Research Center for Computer Science at Schloss Dagstuhl, Germany. Due to its cosy atmosphere and self-contained organization, the center provides an excellent venue for such a school, where informal meetings of participants among each other and with the lecturers are as important as the scientific program.

The School had about 55 participants, one third of them external to the network, plus 14 lecturers. The concept of the school has been to provide tutorial-like introductions to fundamental solutions for

the meta-level tasks M1 – M4, to present examples of applications to specific languages and problem domains, and to offer the opportunity for practical experience in applying techniques and tools in sample projects. With this aim, four different kinds of lectures were offered.

1. *Introductory Lectures* in the form of a keynote by Jean Bézivín and an opening address by Gregor Engels did set up the general motivation and problem space, organized the solution domain according to the meta-level objectives into categories M1 - M4, and gave some background on the SegraVis project as a whole. A mini tutorial on graph transformation by Reiko Heckel provided technical background required by the following lectures.

2. At *Meta-level Tutorials* attendees were introduced to the meta-level solutions for defining and implementing visual modeling techniques, providing a systematic discussion of problems and requirements and a survey of known solutions. To broaden the view, each topic was covered by (at least) two speakers, respectively presenting the classical point of view as well as the solutions more specific to the approach of the network, i.e.

   M1: Uwe Kastens and Mark Minas on *Syntax and Well-Formedness*

   M2: Peter Mosses and Hans-Jörg Kreowski on *Operational and Denotational Semantics*

   M3: Mauro Pezzé, Luciano Baresi, and Reiko Heckel on *Model-based Testing and Analysis*

   M4: Andy Schürr and Martin Große-Rhode on *Integration, Refinement, and Transformation*

3. *Exercises* have been offered on a daily basis, following the topics of the tutorials. Under the guidance of the lecturers, attendees could choose among different techniques and tools to solve the respective problems. A running example, prepared by Gabi Taentzer and Reiko Heckel, has been used throughout. Detailed solutions will be discussed during the Workshop on Graph-based Tools at ICGT 2004 in Rome.

4. *Tool Presentations* have covered tools and meta tools for visual modeling techniques provided by lecturers and attendees. A corresponding survey has been given on the first day by Gabi Taentzer.

5. As *Scientific Highlights*, invited speakers of the community have offered high-level scientific talks on interesting applications of visual modeling techniques and graph transformation to important and solid research areas in computer science, i.e.

   - András Pataricza on *Modeling and Transformations: An Engineering View*
   - Hartmut Ehrig on *Transformation of Petri Nets*
   - Francesco Parisi-Presicce on *Specification and Analysis of Security Policies*

4 May we introduce to you

Finally, let us introduce in more detail four of the 12 sites of the network with their key researchers, topics, and projects. To date, all four groups have positions for research training grants available. Please contact them if you are interested.

Paderborn

The SegraVis team at the University of Paderborn (Germany) consists of two research groups: The Database and Information Systems Group led by Gregor Engels and the Software Engineering Group headed by Wilhelm Schäfer. Both groups have considerable experience in graph transformation systems, their theory, semantics, implementation, and analysis, as well as their application in the context of UML-based software development.

Gregor Engels is the coordinator of the network and with Reiko Heckel, the network manager, co-organizer of the School (see above). The group specializes in operational semantics, consistency and transformation of UML models, model-based testing, stochastic and timed methods, as well as in applications to software architectures, mobility, multimedia applications and Web services.

The Software Engineering Group of Wilhelm Schäfer is well-known for its work on implementing graph transformation in the UML CASE tool Fujaba and applying it in the design of safety critical and real-time systems, as well as in high-school courses on object-oriented programming. Jointly the two groups contribute to all objectives of the network.

The university is host to the International Graduate School of Dynamic Intelligent Systems run by the departments of Computer Science, Mathematics, Business Computing, Electrical and Mechanical Engineering as a recognized center of excellence in this field. Due to its international participants, which are integrated in the different research groups, the graduate school creates a cultural (and linguistic) mix that visitors on SegraVis grants will find most inspiring.

Antwerp

At the University of Antwerp (Belgium), two groups involved in the network: the group on Formal Methods in Software Engineering (FOTS), headed by Dirk Janssens, and the Lab on Software Reengineering (LORE), headed by Serge Demeyer. The FOTS group is interested in the use of techniques based on graphs and graph rewriting in the software development process, in particular model transformation and formal semantics, and the LORE group has expertise in reverse- and reengineering techniques applicable to large scale-software systems. There is a close cooperation between both groups, which jointly consist of about 10 researchers.

Within SegraVis the focus of the team is on the development of languages supporting transformations between software models, including the integration of the proposed formalisms with other models (e.g. UML models), CASE tools (e.g. Fujaba) and standards (e.g. MOF). Within this topic the team concentrates on two problem areas: on the one hand, it investigates the problem of specifying, in a way that is as much as possible platform-independent, the transformations needed for model-driven development, while at the same time supporting the process of generating concrete, and hence platfom-specific, code. On the other hand, the use of rule-based and in particular graph rewriting-based techniques as a basis for a visual language for refactoring is investigated: refactorings are transformations intended to improve the structure of a system, while preserving its external behavior. Since these transformations are mostly local and triggered by the occurrence of certain patterns, graph transformation is a natural candidate for their formal description. An international workshop on the use of graph rewriting for refactoring was held in Antwerp in April 2004.

Besides SegraVis, FOTS and LORE are involved in a number of projects concerning reengineering and model-driven as well as aspect-oriented software development. As a result, the research of these groups is relevant for a broad range of the SegraVis are objectives, and offers an interesting and productive environment for prospective grant holders.

Barcelona

The SegraVis team at Barcelona consists of two groups: The Algebraic Methods Group lead by Fernando Orejas and the group on Synthesis and the Verification of Concurrent Systems headed by Jordi Cortadella.

The group of Fernando Orejas has long experience in the use of these of algebraic and transformation-based methods for defining the semantics of specification languages. The current interests of the group in relation with the network objectives are, on one hand, in the semantic integration of modeling formalisms and, on the other, in the study of component based software modeling and design.

The group of Jordi Cortadella has a strong background in the development of methods and tools for the synthesis and verification of (hardware) systems. A well-known tool developed by the group is PETRIFY, a CAD tool for the synthesis of concurrent systems modeled by Petri Nets. Currently, the main interest of the group is in the verification of timed systems using abstract interpretation techniques.

Fernando Orejas has been the local organizer of ETAPS 2004, the Joint European Conference on Theory and Application of Software which was held in Barcelona. The city itself is certainly one of the most beautiful in Europe, with a lot of potential for research in unrelated subjects, like architecture and cuisine.

Bremen

The SegraVis team at the University of Bremen (Germany) comprises two research groups: The Group on Theoretical Computer Science (TCS) headed by Hans-Jörg Kreowski and the Database Systems Group led by Martin Gogolla.

The TCS group specializes in language-independent modeling of systems using rule-based graph transformations. This generic framework integrates existing approaches and allows for a uniform view on model transformations. In this context, transformation units as a key concept for modularity and reuse of rule-based systems are developed and applied, e.g., to define formal semantics of parts of the UML. Further important research topics are the modeling of software agents and autonomous logistic processes, syntactic methods of picture generation, algebraic specification, and DNA computing.

The Database Systems Group applies graph transformation in the context of UML-based software engineering. For example, an integrated formal semantics has been developed for the mainstream UML diagrams. The group also investigates formal specification of safe and secure systems, semantics of data base languages, formal methods in information systems design, object-oriented software development in general, and software metrics.

Both groups are associated with the Bremen Institute of Safe Systems (BISS), which aims at increasing the safety and reliability of software and embedded systems by formal methods, and the Center for Computing Technologies (TZI), the objectives of which are the research, development, and transfer of innovative computing technologies. The TCS group contributes as a member to the Logistics Research

Institute (FoLo) and the Collaborative Research Centre on Autonomous Cooperating Logistic Processes (CRC 673). Moreover, the TCS group hosts the Competence Center for Women in Science and Engineering at the University of Bremen, and Hans-Jörg Kreowski currently chairs the Forum Computer Professionals for Peace and Social Responsibility (FIfF).

ETAPS 2004

Hartmut Ehrig
Technical University of Berlin

The European Joint Conferences on Theory and Practice of Software were held this year in Barcelona from March 27-April 4, 2004. Actually ETAPS 2004 was the seventh instance of ETAPS established 1998 in Lisbon followed by ETAPS-conferences in Amsterdam (1999), Berlin (2000), Genova (2001), Grenoble (2002), and Warszawa (2003). This year ETAPS was breaking again the records of previous years concerning the number of submitted papers for the 5 main conferences (600), the number of satellite events (25) and the total number of participants (725). The organisation of ETAPS 2004 by Fernando Orejas (Chair) Jordi Cortadella (Satellite Events) and their team was really outstanding concerning the support for the scientific and also the social program. In addition to the traditional ETAPS dinner which took place in the famous Cordorniu Wineries, four workshop dinners and two receptions were offered. Even for the only free evening on Tuesday Fernando was able to organise 300 tickets for the fabulous Flamenco performance “A Poet in New York” which were given to the participants almost for free (5 Euro instead of 27 Euro). Due to a request of the ETAPS steering committee to maximise the attendance of the participants in the lectures the organisation team was even able to change the weather conditions from sunshine to rain in order to protect people from the temptation of sightseeing during lecture time.

Concerning the scientific part the chairs of the main conferences (FASE, ESOP, TACAS, FOSSACS, CC) and the organisers of the satellite events were very successful to select a high quality program, supported by about 350 members of all the program committees. Thanks also to the ETAPS steering committee with Chairman Jose Luiz Fiadeiro for the overall guidance of the ETAPS conferences. In fact, ETAPS has turned out to be even more successful than we have dreamed 10 years ago, when we started the discussion to join the predecessors TAPSOFT, ESOP, CC and TACAS to the new joint conference ETAPS. One of the main scientific highlights at ETAPS 2004 was the invited lecture by Robin Milner with the title “A Grand Challenge: Theories for Global Ubiquitous Computing”. Robin reported about a recent conference in the UK concerning grand challenges in Computer Science for the next 15 years and especially about his view of theoretical foundations for global ubiquitous computing.

Following the tradition of previous ETAPS conferences the three European Associations EATCS, EAPLS and EASST supporting ETAPS have offered an award for the best paper of the five main conferences in the corresponding focus area of each of the associations. The winners (EAPLS: Gogul Balakrishnan and Thomas Reps. Analyzing memory accesses in x86 executables, EASST: Marieke Huisman, Dilian Gurov, Christoph Sprenger and Gennady Chugunov. Checking Absence of Illicit Applet Interactions: A Case Study, EATCS: Daniel Kirsten. Distance Desert Automata and the Star Height One Problem) received their award during the ETAPS dinner on Wednesday together with some gifts from Fernando as ETAPS-organiser. Moreover, Bernhard Steffen was honoured by the TACAS community for 10 successful years of TACAS, starting as workshop at TAPSOFT 1995 in Aarhus.

The next ETAPS conferences will be organised by D. Sannella in 2005 in Edinburgh and by J. Knoop in 2006 in Vienna.

Software Engineering Excellence in Barcelona

Gruia-Catalin Roman
Washington University in Saint Louis
U.S.A

As one of the two invited speakers of the *7th International Conference on Fundamental Approaches to Software Engineering (FASE 2004)*, one of the ETAPS 2004 conferences, I found myself in the enviable position of promoting my ideas to a large and exceptionally bright audience that included highly recognized leaders in the field. It was indeed a thrill to recognize so many faces and a humbling feeling to know that such intellectual strength will be judging every idea one puts forth. ETAPS, in general, and FASE, in particular, have been able to attract this year not only a record number of submissions, but also a diverse, high energy, and demanding set of participants both from Europe and from across the globe. As is the case with other prestigious venues, in this case the "European" label has become mostly an acknowledgement for the origin of an event that has a truly global reach.

Michel Wermelinger and Tiziana Margaria-Steffen, along with the program committee and the still broader set of reviewers, assembled together a three-day program that was impressive along several important dimensions: quality, scope, and relevance to modern software engineering. The notion that current software engineering research and development must be application and user centered emerged as a strong undercurrent for much of the work being presented in the meeting. Both invited lectures set the tone in this respect. The presentation on distributed information management (Abiteboul) highlighted the special role service provision plays in today's delivery of applications across the Web; the presentation on context awareness (Roman) sought to emphasize the importance of employing formal methods in applications that entail physical and logical mobility.

Within the setting of the regular conference program, the need for being highly sensitive to application needs dominated the session dedicated to "smart cards." Concerns of critical importance to the end user, security and integrity, figured prominently both in the presentations and the discussions they engendered. The thread having to do with access rights

and controls extended to still another session, which also revisited the theme of service provision present in one of the invited papers. The importance of formalizing user requirements was evident both in the sessions centered directly on applications and in a series of reports on modeling and requirements specification. Formalization and proper tool support emerged here as a common and dominant theme, despite the wide ranging set of technical approaches.

Software architecture tied together three separate sessions. Presentations related to objects and aspects emphasized qualitative elements in the evolution of the software architecture while component composition assumed a dominant presence in two separate sessions. Treatment of this topic appeared to resonate strongly with the needs of modern software development practice, even though the perspectives adopted by the various presenters varied greatly. Testing and analysis completed a picture that offered coverage of the entire software development life cycle. Reports on testing methods explored ways to leverage off assertions and architectural specifications while model checking, as an effective analysis technique, dominated the analysis session with some attentions being paid to static analysis as well.

Convergence with regard to the dominant research themes and broad coverage of the multitude of facets characterizing software engineering practice suggests a high degree of intellectual cohesion among the participants. This was indeed surprising to see in a setting that exhibited such great diversity of methods, experiences, and perspectives on the field. A special session on wild ideas could complement nicely next time such a program is assembled together.

Integration of Specification Techniques for Applications in Engineering 1998-2004

Hartmut Ehrig *
*Institut für Softwaretechnik und Theoretische Informatik
Technische Universität Berlin, Germany
Email: ehrig@cs.tu-berlin.de

In 1998 the German Research Council (DFG) started the Priority Program "Integration of Software Specification for Applications in Engineering", short SoftSpez [EG01]. The main aim of SoftSpez is on one hand the integration of software specification techniques for different aspects of a system and on the other hand the integration of formal techniques with semiformal techniques from engineering. Two major reference case studies have been defined in the area of traffic control systems and in production automation in order to show how these techniques can be used to solve typical problems in engineering.

The Priority Program SoftSpec was initiated by W. Brauer, M. Broy, H. Ehrig, H.J. Kreowski, H. Reichel and H. Weber concerning different aspects from Computer Science and E. Schnieder and E. Weskämper concerning the two application areas in engineering. After acceptance of SoftSpec by the German Research Council for the period 1998-2004 a call for specific projects within this priority program was launched, where 11 projects from about 75 project proposals were accepted for a period of 2 years. Since 1998 each year the main research proposals and results of the projects have been presented on an annual colloquium of the priority program, and each two years the projects have been evaluated by an independent group of referees appointed by the German Research Council.

The cooperation between the projects has been organized in different focus areas with several focus area meetings since 1999. In addition to the annual colloquia and the focus area meetings on a national level also three international workshops have been organized by the Priority Program SoftSpec in order to present the concepts and results to the international scientific community and to get feedback from international experts as invited speakers for these workshops. The first and second international workshop INT 2000 and INT 2002 on "Integration of Specification Techniques for Applications in Engineering' have been launched as satellite events of ETAPS 2000 in Berlin resp. ETAPS 2002 in Grenoble, where the international forum of ETAPS, the "European Joint Conferences on Theory and Practice of Software" is most important. The third international workshop INT 2004 was organized this year by H. Ehrig, F. Orejas, A. Pataricza, W. Reif and G. Schröter as satellite of ETAPS 2004 in Barcelona.

The highlights of this workshop were the invited lectures by D. Harel and D. Björner and the talks by W. Reif and M. Große-Rhode on main topics of the Priority Program.

D. Harel presented his novel approach to programming of reactive systems, in which inter-object scenario-based behavioral requirements are "played in" directly from the systems GUI, and behavior

can then be “played out” freely adhering to all the requirements. The language he used is an enriched version of the LSCs (live sequence charts), that were originally developed by him and W. Damm [DH01]. A detailed presentation of this approach is given in his book [Har03]. Verification of LSCs on the other hand is one of the main topics of the USE project of W. Damm in the DFG Priority Program [DW03].

In his lecture “UML-ising Formal Techniques” D. Björner reported on ways and means of integrating formal techniques such as RAISE, Timing, Duration Calculus, LSCs, Petri Nets, Statecharts and ER-diagrams. This integration covers - similarly to UML - a wide spectrum of specification techniques and can especially applied to the area of railway systems [Bj03] which is one of the application areas of the Priority Program. Within the SoftSpez project ForMoSA W. Reif developed an integrated approach for safety analysis of critical, embedded systems. In his presentation at INT 2004 he reported about the application of his approach to the traffic control case study, which is another important application area within SoftSpez, and especially the height control of the Elbtunnel in Hamburg [OR03].

One of the main theoretical results of the DFG Priority Program was presented at INT 2004 by M. Große-Rhode. It is the language- and method-independent model integration in the SoftSpez project IOSIP. This semantical framework for model integration is based on a uniform concept of transformation systems, which allows the semantic integration of heterogenous software specifications [Gro03].

The final meeting of the Priority Program will be the 2004 colloquium of SoftSpez on September 27/28, within the frame of the “DFG-Wissenschaftssommer” organized by E. Westkämper in Stuttgart.

A documentation on “Integration of Specification Techniques for Applications in Engineering” will appear as a Springer LNCS volume, including on one hand main results of the Priority Program SoftSpec in Germany and on the other hand contributions of other international experts in this field.

Literatur

[Bj03] D. Björner. 2003. Dynamics of Railway nets: On an Interface between Automatic Control and Software Engineering. Proc. CTS 2003 (10th IFAC Symp. on Control in Transportation Systems).

[DH01] W. Damm, D. Harel. 2001. LSCs: Breathing Life into Message Sequence Charts. Formal Methods in System Design 19(1) pp.: 121–141.

[DW03] W. Damm, B. Westphal. 2003. Live and let die: LSC-based verification of UML-models. Springer LNCS 2852.

[EG01] H. Ehrig, M. Große-Rhode. 2001. Integration von Techniken der Softwarespezifikation für ingenieurwissenschaftliche Anwendungen. Inf. Forsch. Entw.16 pp.: 110–117.

[Gro03] M. Große-Rhode. 2003. Semantic Integration of Heterogenous Software Specifications. In EATCS Monographs in TCS, Springer.

[Har03] D. Harel, R. Marelly. 2003. Come, Let’s Play: Scenario-Based LSCs and the Play-Engine. Springer.

[OR03] F. Ortmeier, W. Reif et al. 2003. Safety Analysis on the Height Control System for the Elbtunnel. Reliability Engineering and System Safety 81(3), pp.: 259–268.

Report on the Workshop on Formal Foundations of Embedded Software and Component-based Software Architectures (FESCA)

Ralf H. Reussner [1] * and Juliana Küster-Filipe ** and Iman H. Poernomo * and Sandeep Shukla *****

*University of Oldenburg / OFFIS, Germany
**University of Edinburgh, UK
***Monash University, Australia
****Virginia Tech, USA

Abstract. *The FESCA 2004 workshop took place on April, 3rd, 2004 as a satelite event of ETAPS in Barcelona. Its aim is to bring together researchers from academia and industry interested in formal modelling as well as associated analysis and reasoning techniques with practical benefits for embedded and component-based software engineering. Nine papers were accepted for presentation. Two guest speakers, Manfred Broy (TU München, Germany) and Constance Heitmeyer, Naval Research Lab, USA) contributed to a highly successful event. Besides the discussion of the presented approaches, several problems of integrating scientific results of formal methods into industrial software engineering practice were discussed.*

Keywords: formal methods, CBSE, embedded systems, knowledge transfer

1 FESCA's Motivation and Background

The aim of the FESCA workshop is to bring together researchers from academia and industry interested in formal modeling approaches as well as associated analysis and reasoning techniques with practical benefits for embedded software and component-based software architectures.

Component-based software design has received considerable attention in industry and academia since object oriented software development approaches became popular. Recent years have seen the emergence of formal and informal techniques for the specification and implementation of component-based software architectures. With the growing need for safety-critical embedded software-systems, this trend has been amplified. Various component models, models of computation and component composition techniques and frameworks have been proposed in literature.

[1]Corresponding author: Software Engineering Group, Department of Computing Science, University of Oldenburg / OFFIS, Escherweg 2, D-26121 Oldenburg, Germany, reussner@informatik.uni-oldenburg.de

An underlying theme of all these is to establish correctness or correct by construction software engineering. It is widely believed that formal methods are a good means to achieve correctness in complex software systems. However, there are several reasons, why results of academic research in formal methods have not contributed to the practice of component software and embedded systems construction as much as expected:

1. A formal specification language alone is not an asset for itself in industrial software development. Although a formal specification of the system to be built makes developers to think on the system prior to its implementation (which helps to understand the system), it is a weak justification for applying formal methods. This is because it is unclear whether semi-formal or informal techniques would not have had the same effect on system understanding. In particular, the benefit of the formal specification's precisely defined semantics is not used. Therefore, we argue, that formal methods are only justified if they enable system analysis not possible without formal specifications. Very importantly, these analysis has to be tool supported and therefore has to be done automatically.

2. Advances in formal methods and formal verification do not scale up with the increasing complexity of software. In fact, software complexity is increasing at a pace greater than Moore's law due to the use of software in many facets of computation such as multi-media, real-time systems, networking protocol stacks, complex/real-time operating systems, networked systems software, pervasive networking, etc. Hence, it is not only the sheer size of the software hindering the application of formal methods, but also a lack of specific constructs to model above-mentioned systems.

(Note, that additional reasons identified during the discussions are summarised in section 3.)

Therefore, FESCA called specifically for contributions (a) dealing with analysis techniques giving an immediate benefit to formal specifications, and (b) for specification techniques dealing with the specification of novel computational environment. Here we concentrated on techniques for specifying embedded software-systems and software making use of current middleware platforms (such as OMG's CORBA or implementations of Sun Microsystems's EJB or Microsoft's .NET). There are several reasons for joining this usually distinct areas of software engineering (embedded systems and middleware-based systems (usually for enterprise application)):

- component-orientation is a cross-cutting general principle, not only an implementation reuse technique, but also on higher levels of abstraction, component-orientation is promising (e.g., compositional reasoning [dRdBH+01], establishing trust by trusted components and prediction models [HMSW02]). Therefore, also embedded systems are expected to benefit from component-orientation on several levels of abstraction.

- As embedded system software controls its technical environment most often this software is safety critical. Regardless, whether components are in use or not, this demand for safety is a good argument for investing in the use of formal methods.

- The underlying formalisms for modelling embedded systems or components are widely shared, such as MSCs [C. 92], LSCs [DH01], state-based approaches (e.g., statecharts [Har87]), or petri-nets [Pet62].

Therefore, the organisers find the widened scope of FESCA beyond common communities boundaries particularly fruitful for each side. In particular, we find the joint exploration of shared formalisms for reasoning and prediction for domain-specific problems very promising. Due to this and the given success of this FESCA workshop, we are especially happy to announce that FESCA will be continued at the next ETAPS 2005 in Edinburgh, UK.

The success of FESCA shows in the high quality of the submissions attracted. Common topics of the seven papers presented where formalisation of middleware, architectural reconfiguration, performance and system integration. (Two accepted submissions could not be presented due to lacking travel funding. However, the papers can be found on the FESCA web site.) Besides these presentations, FESCA gained attraction through our two invited speakers Manfred Broy and Constance L. Heitmeyer.

2 Abstracts of the Contributions

In the following summaries of the presented papers are given. Full papers can be found on the FESCA web-page `http://www.csse.monash.edu.au/fesca/`.

Invited Talk I: Manfred Broy, TUM, Germany: Time, Abstraction Causality, and Modularity in Interactive Systems

The invited talk discussed discrete models of interactive distributed systems structured into components and operating concurrently in a time frame. For such models of the data or signal flow in interactive system we assume that there is a source and a cause for each communication event and its associated information. To understand the logical dependencies for the events of systems causality is a key issue for reasoning about the event flow. Being interested in a structured modular approach we want to be able to abstract away all internal aspects of systems that are used as components within a system's architecture. We speak of interface abstraction. The interface abstraction is to keep only the aspects relevant for the usage of the com- ponent and the construction of the interface abstraction of the architecture. We speak of modularity if the interface abstraction of an architecture is the result of the composition of the interface abstractions of all its components. In particular, we discuss and study the relationship and dependencies between causality, input and output, compositionality, and the granularity of time.

Session I – Middleware-based Systems

Formal Modeling Of Middleware-based Distributed Systems by Arnab Ray and Rance Cleaveland (SUNY at Stony Brook, USA)

Effective design of middleware-based systems requires modeling notations that allow the use of process-interaction schemes provided by dierent middleware packages directly in designs. Traditional design notations typically only support a fixed class of interprocess interaction schemes, and designers wishing to use them for modeling middleware-based systems must devote signicant effort to encoding the middleware primitives in the notation. In this paper, we demonstrate how a new graphical design notation, Architectural Interaction Diagrams (AIDs), which provides parameterized support for different

interaction schemes, may be used to model a real-life middleware-based system like the Event Heap coordination infrastructure of the i-Room ubiquitous computing environment.

A Generic Framework for Connector Architectures based on Components and Transformations **by H. Ehrig, J. Padberg, B. Braatz, M. Klein (Technical University Berlin, Germany) and F. Orejas, S. Perez, E. Pino (Universitat Politecnica de Catalunya, Barcelona, Spain)**

The intention of this paper is to extend our generic component framework presented at FASE 2002 to a specific kind of connector architectures similar to architectural connections in the sense of Allen and Garlan. In our generic component framework we have considered components with explicit import, export and body parts connected by embeddings and transformations and composition of components with a compositional transformation semantics. Our framework, however, was restricted to components with a single import and export interface. Here we study architectures based on connectors with multiple imports and components with multiple exports. Architectures studied in this paper are built up from components and connectors in a non-circular way. The semantics of an architecture is defined by reduction step sequences in the sense of graph reductions. The main result shows existence and uniqueness of the semantics of an architecture as a normal form of reduction step sequences. Our generic framework is instantiated on one hand to connector architectures based on CSP as the formal specification technique in the approach by Allen and Garlan. On the other hand it is instantiated to connector architectures based on high-level-replacement systems in general and Petri nets in particular. A running example using Petri nets as modeling technique illustrates all concepts and results.

Session II – Architecture Reconfiguration

Hierarchical Temporal Specifications of Dynamically Reconfigurable Component Based Systems **by Nazareno Aguirre (Universidad Nacional de Rio Cuarto, Cordoba, Argentina) and Tom Maibaum (King's College London, UK)**

We study how temporal specifications of reconfigurable component based systems can be hierarchically organised. We do so by extending a previously introduced declarative prototypical language to admit the definition of hierarchical subsystems. Each subsystem has an internal architecture, composed of its internal interacting (simpler) subsystems, and basic components. The internal architecture of a subsystem can change at "run time" by means of reconfiguration operations. The notion of subsystem provides an extra coarse grained unit of modularisation, that complements that of components. Since component interaction is achieved by means of coordination, a component or subsystem can be represented by a logical theory isolated from the rest of the system. This, in combination with the possibility of hierarchically organising a specification, has a special impact in reasoning, since it allows us to further localise the proof efforts to the relevant subparts of a specification.

Generating snapshots of a component setting **by Sotiris Moschoyiannis (University of Surrey, Guildford, UK)**

Software components are often seen as panacea when faced with the challenges of the increasing use of software in many diverse areas of computation. However, the complex 'call interplay' at the interfaces between components often results in pathological behaviour and hinders effective reuse. There is a clear need for languages for documenting and specifying components in such a way that insulates them from changes in the configuration. In this paper we describe the use of Live Sequence Charts (LSCs) to describe component interactions. Then, we advocate an approach for formalising those interactions to ensure that components interact properly whilst making minimal assumptions about their neighbours.

Invited Talk II: Constance L. Heitmeyer, NRL, USA: Managing Complexity in Software Development with Formally Based Tools

Over the past two decades, formal methods researchers have produced a number of powerful software tools designed to detect errors in, and to verify properties of, hardware designs, software systems, and software system artifacts. Mostly used in the past to debug hardware designs, in future years, these tools should help developers improve the quality of software systems. They should be especially useful in developing high assurance software systems, where compelling evidence is required that the system satisfies critical properties, such as safety and security. This paper describes the different roles that formally based software tools can play in improving the correctness of software and software artifacts. Such tools can help developers manage complexity by automatically exposing certain classes of software errors and by producing evidence (e.g., mechanically checked proofs, results of executing automatically generated test cases, etc.) that a software system satisfies its requirements. In addition, the tools allow practitioners to focus on development tasks best performed by people – e.g., obtaining and validating requirements and constructing a high-quality requirements specification.

Session III – Optimization, Performance and Integration

Formally Specifying Dynamic Data Structures for Embedded Software Design: an Initial Approach **by E. G. Daylight, B. Demoen and F. Catthoor (IMEC and K.U. Leuven, Belgium)**

In the embedded multi-media community designers deal with data management at different levels of abstraction ranging from abstract data types and dynamic memory management to physical data organisations. In order to achieve large reductions in energy consumption, memory footprint, and execution time, data structure related optimizations are a must. However, the complexity of describing and implementing such optimized implementations is immense. Hence, a strong practical need is present to unambiguously (i.e., mathematically) describe these complicated dynamic data organisations.

The objective of this article is to formally describe data structures and access operations -or dynamic structures for short- that we have implemented in prior , application related work. We do this by (a) extending the syntax and semantics of Separation Logic -a logic developed recently in the program verification community- and (b) using it as a specification language for our applications.

The short-term benefit of this work is that it allows the embedded software designer to unambiguously

express and hence more easily explore low cost, dynamic data structures. In practice this means that the designer can clearly reason and consequently implement nontrivial but optimal dynamic data structures. The benefit in the long-term is that it provides an avenue for future optimizing compilers to increase the global scope of optimizations that are related to dynamic data management.

Model based development and integration of embedded components: an experience report from an industry project **by Martin Grosse-Rhode and Stefan Mann (Fraunhofer ISST, Germany)**

An experience report on the introduction of a model based method for the development and integration of component-based embedded software in the automotive industry is given. Although the method itself is not formal, formal techniques were used in the development of the method and are visible at the meta level. The aim of this report is to improve the coordination of theoretical achievements and industrial needs, and gain realistic ideas on the dierent roles formal methods for component based software development could play in larger industrial applications.

Towards Performance Evaluation of Component Based Software Architectures **by Viktoria Firus and Steffen Becker (University of Oldenburg, Germany)**

This paper reviews different approaches for predicting the performance of software architectures while keeping component based systems in mind. For these approaches we present classifying properties. We investigate formal analytical prediction models and execution based simulation models. In particular, we discuss network queuing models. As a result, we show the benefits and limitations of these models for predicting performance with respect to component based software architectures. Finally, we sketch future work on a mixed approach for component based performance evaluation, utilising both, simulations and predictions.

3 Discussion: Problems of Transferring Academic Research into Industrial Practice

Discussion sessions partially dealt with the topic of the respective talks. Besides, another topic recurrently came up during several sessions: the problem of transferring formal methods from academia into practice. Given that this was one of the central topics of FESCA, we extend the discussion of the workshop by some afterthoughts in this report. Besides the generally accepted need to make formal methods applicable, several reasons were discussed, hindering the transition of formal methods into practice:

1. **Many formal methods do not scale:** Developed in an academic world, formal methods are applied to rather limited toy examples. Opposed to that, major challenges in industry relate are originated by handling large and complex systems. Unfortunately, the costs of applying formal methods to large systems most often grow super-linearly.

2. **Insufficient constructs to model the real world:** Increasingly, computer systems use concepts beyond the standard functional model (inputs are mapped by computations to outputs). Examples

are distribution, concurrency, mobility, dynamic reconfiguration and ubiquity. Some of these concepts are long running topics for research (such as concurrency) while modelling and analysing others are still challenging. However, as between the practical motivation and the actual research result some assumptions and simplifications have to be made to make the problem tractable, the research result not necessarily solves the original real-world example, but inevitably focuses on specific aspects.

3. **No time for validation:** The development of a formal method is a task for itself. Given the usual time frame of academic dissertations, for the validation of the method often no time is left.

4. **Unclear practical benefit:** Academic research is often motivated by intriguing mathematical questions. In principle this is justified, because one hardly knows what comes out of research, hence one cannot exclude in principle the value of such motivated research. However, the outcome of such research often has no practical benefits and lies much more in a deeper understanding of the problem. If such research is later for the sake of better selling motivated by practical questions, one has the impression of ”solution found, in search for a problem”. Obviously, such methods do not really solve real world problems with their many different facets.

5. **Missing tool support:** Scientific progress is considered as adding scientific knowledge to a given area of research. The implementation of tools does not create scientific knowledge, at the best it create experience how to apply software development techniques. Due to that, for many scientists no motivation exists to invest time in the demanding task of tool creation. However, without tool support, the application of most formal methods does not scale up (see reason 1).

6. **Lack of industrial experience for academics:** Many academics spend their life-time in universities but have not worked in industry or in industry cooperation projects. Consequently, researchers provide a deep understanding of mathematical techniques for solving problems, but may have a rather limited knowledge on ”real” problems. In particular, real problems are multi-faceted, including non computer science issues. Interestingly, single aspects are often rather trivial, annoying researchers having solutions to this single aspect. The complexity of industrial systems often simply stems from their sheer size while the single algorithms and data-structures are simple.

7. **Up-front costs in industrial processes:** Formal methods, like nearly all software engineering techniques, increase the up-fond costs of software development. Additional effort has to be spend during the specification or design of the product, while hoping to save costs during implementation, testing and maintenance. Generally, this is problematic: (a) Even if the overall savings effort of a technique is demonstrated several times, (e.g., in case of software reviews and inspections [Gla02]), managers are often reluctant to deploy these techniques. This is because, managers often still consider the implementation step as the most important one and do not see the project’s progress if effort is spent in steps prior to the implementation. (b) Due to a lacking validation in industrial contexts, the saving effects of many formal methods is unclear or at least not quantified. This results in a hen-egg problem: As long a formal method is not deployed, it will not be deployed.

8. **Goals of research differ from industrial needs:** Probably the most fundamental reason beyond all previously mentioned ones is that the motivation and credit one gets for work fundamentally differs between academic researchers and industrial software developers. Industry projects have to succeed under tight time and cost constraints. Sometimes, time-to-market is of higher concern than the quality of the first version shipped. Success is measured in producing the product with the lowest amount of these constrained resources. The application of a not-yet validated formal method (having the only guaranteed property of increasing up-front costs) is not very appealing. On the academic side, the contribution of research is measured not only in soundness but also in originality and novality. This is done for good reasons, but often has the drawback, that the work on transitioning existing formal methods into practice (such as tool construction, adapted process models or case studies) are still considered as less scientific compared to the development of new formalisms. As new formalisms have to add something to the old ones (which are also not yet understood and deployed in industrial practice), the gap between research and practice will be widened.

We intentionally omitted the training aspect. Of course, the specific knowledge required to deploy a formal method is most often not present at industrial developers and the awareness of new formal techniques and their benefits is often low. This was usually considered as a problem to be solved by increasing the number of software developers having an academic degree in software engineering or computer science. While this is partially true, the above list of problems still is valid when having teams with academically trained staff. Although the internet-boom is over, current industrialised countries have a lack of academically trained software developers (still or again). As the demand of software professionals still exceeds the capacities of universities, this situation will not change in the near future. Consequently, the transition of formal methods in industrial practice can not solely depend on trained people (and, as argued above, even that would not solve the above mentioned list of problems). Interestingly with respect to the education, differences between the US and Europe exist: In Europe formal methods education has been quite popular in academic programs (at least until now). In the US only a handful schools have experts in formal methods in their software or hardware engineering programs. Most schools have curriculi that does not include anything about formal methods or its benefits in system design. This needs to change drastically if the benefits of formal approaches are to be understood and utilized in the industry. As a result, those few companies using formal methods focus primarily on making their model checking or test generation engines more efficient without substantially concentrating on the methodology aspects of hardware design where formal methods need to play an important role. However, to change this, we need to create curriculum that teaches students to focus on formal methods based design methodologies.

However, it is important to realise that these problems hinder the adoption of academic research in industry but the adoption is not made impossible. There are successful ways to overcome these problems. Like Robin Milner in his invited ETAPS talk, the participants of FESCA also advocated for close cooperation projects between industrial professional and researchers. These joint projects result in deeper understanding of the nature of real world problems by academics and the power of formal methods by practitioners. In particular, challenges and benefits of applying a novel method become clear on both sides. However, this also means change in the value systems: on the one hand, solving the problems of making an existing formal methods applicable should be a contribution for a researcher of itself, on

the other hand, switching to long-term views and thinking strategically in longer time scales is usually beneficial for companies in general. In the US, Intel has been using formal verification for over a decade now, in Pentium III floating point unit, various units of Pentium IV, and the Itanium processor. Some bus specification at Intel are formally verified as well. Microsoft is using static driver verification in windows drivers projects. MS Research has made efforts to build formal methods expertise. Other than that not many companies are doing this. A number of EDA tools in hardware design automation industry have come up in recent times.

Success stories of joint innovative technology transfer projects exist: For example the various institutes of the Fraunhofer society in Germany or the OFFIS research institute in Oldenburg are significantly funded by joint projects. As an example, the joint ViSEK-project (Virtual Software Engineering Competence Centre, www.visek.org) helps to bridge the gap between industry and research. The increasing role of so-called empirical software engineering [Tic98] emphasises the scientific role of empirical validation of newly proposed methods by case studies or even controlled experiments.

4 Conclusions

Future efforts for FESCA are twofold. Firstly, given the high quality of the presented papers we decided to invite the authors to submit extended versions of their contributions to a book volume on Formal Foundations of Embedded and Component-based Systems. Details will be given by a separate call for contributions. Negotiations with appropriate publishers are pending.

Secondly, we are happy to announce that FESCA will be continued next year on ETAPS 2005. In particularly, we plan to schedule the workshop close to the FASE conference as both events are likely to attract participants from similar research communities. Details will be given soon by a call for papers and the FESCA web-page `http://www.csse.monash.edu.au/fesca/`.

List of Participants

Manfred Broy	TUM, Germany
Edgar Daylight	IMEC & KU Leuven, Belgium
Martin Grosse-Rhode	Fraunhofer ISST, Germany
Constance L. Heitmeyer	NRL, USA
Juliana Küster-Filipe	Univ. of Edinburgh, UK
Tom Maibaum	King's College London, UK
Sotiris Moschoyiannis	Univ. of Surrey, UK
Julia Padberg	TU Berlin, Germany
Fernando Orejaz	UPC, Spain
Iman Poernomo	Monash Univ., Australia
Arnab Ray	SUNY at Stony Brook, USA
Ralf Reussner	Univ. Oldenburg / OFFIS, Germany
Jianli Xu	Nokia Research, Finland

References

[C. 92] C. C. I. T. T. CCITT Recomendation Z.120: Message Sequence Chart (MSC92). Technical report, CCITT, Geneva, 1992.

[DH01] Werner Damm and David Harel. LSCs: Breathing life into message sequence charts. *Formal Methods in System Design*, 19(1):45–80, 2001.

[dRdBH+01] Willem-Paul de Roever, Frank de Boer, Ulrich Hannemann, Jozef Hooman, Yassine Lakhnech, Mannes Poel, and Job Zwiers. *Concurrency Verification: Introduction to Compositional and Noncompositional Methods*. Number 54 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge, UK, November 2001.

[Gla02] Robert L. Glass. *Facts and Fallacies of Software Engineering*. Addison-Wesley Pub Co, 2002.

[Har87] D. Harel. Statecharts: a visuel approach to complex systems. *Science of Computer Programming*, 8(3):231–274, 1987.

[HMSW02] Scott A. Hissam, Gabriel A. Moreno, Judith A. Stafford, and Kurt C. Wallnau. Packaging predictable assembly. In *Proceedings of the IFIP/ACM Working Conference on Component Deployment*, pages 108–124. Springer-Verlag, 2002.

[Pet62] C. A. Petri. Fundamentals of a theory of asynchronous information flow. In *Information Processing 62*, pages 386–391. IFIP, North-Holland, 1962.

[Tic98] W. Tichy. Should Computer Scientist Experiment More? – 16 Reasons to Avoid Experimentation. *IEEE Computer*, 31(5):32–40, 1998.

Call for Papers

Ninth International Workshop on Formal Methods for Industrial Critical Systems (FMICS 04)

Linz, Austria, 20-21 September 2004
Co-located with ASE 2004

http://www.fmics04.cclrc.ac.uk/

SCOPE

The aim of the FMICS workshops is to provide a forum for researchers who are interested in the development and application of formal methods in industry. In particular, these workshops are intended to bring together scientists who are active in the area of formal methods and interested in exchanging their experiences in the industrial usage of these methods. These workshops also strive to promote research and development for the improvement of formal methods and tools for industrial applications.

TOPICS

Topics include, but are not restricted to:

- Tools for the design and development of formal descriptions
- Verification and validation of complex, distributed, real-time systems and embedded systems
- Verification and validation methods that aim at circumventing shortcomings of existing methods in respect to their industrial applicability
- Case studies and project reports on formal methods related projects with industrial participation (e.g. safety critical systems, mobile systems, object-based distributed systems)
- Application of formal methods in standardization and industrial forums

IMPORTANT DATES

Deadline for Submissions:	**June 21, 2004**
Accept/Reject Notification:	**July 26, 2004**
Final Manuscript:	**August 23, 2004**
Workshop:	**September 20-21, 2004**

SUBMISSIONS

Papers submitted to FMICS 04 must be in English and present original research that is unpublished and not submitted for publication elsewhere. Papers should be between 10 and 16 pages, with a clear abstract and list of keywords, formatted according to the ENTCS guidelines (http://math.tulane.edu/~entcs/). The proceedings of the workshop will be published physically as technical reports of the Johannes Kepler University and electronically in Electronic Notes in Theoretical Computer Science. Extended version of best papers will be invited for publication in an special issue of a high quality international journal.

PROGRAM COMMITTEE

Alvaro Arenas (CCLRC/RAL, UK)
Thomas Arts (IT-Univ. in Gothenburg, Se)
Gilles Barthe (INRIA Sophia-Antipolis, Fr)
Juan Bicarregui **co-chair** (CCLRC/RAL, UK)
Armin Biere (ETH Zürich, Ch)
Lubos Brim (Masaryk Univ., Cz)
Andrew Butterfield **co-chair** (Dublin Univ., Ie)
Muffy Calder (Univ. of Glasgow, UK)
Wan Fokkink (CWI, Nl)
Maria del Mar Gallardo (Malaga Univ., Es)
Leszek Holenderski (Philips, Nl)
Diego Latella (CNR/ISTI Pisa, It)
Martin Leucker (Uppsala Univ., Se)
Radu Mateescu (INRIA Rhone-Alpes, Fr)
Ina Schieferdecker (Fraunhofer FOKUS, De)

European Association of Software Science and Technology EASST

Who are we?

EASST is a European non-profit Association that aims at promoting research, development and applications in the area of systematic and rigorous engineering of software and systems.

What are our aims?

Software and Systems Engineering does not receive the public recognition it deserves as one of the most advanced technologies with a great impact on Europe‘s economic and societel prosperity. This is due to a large extent to the low degree of visibility of the community. Especially research is scattered around a rather large number of communities, meetings in different conferences and workshops.

How do you benefit?

When joining us you enter a larger community and you will help to strengthen a new association that is aiming at a better visibility and recognition of your work.
When joining us you will benefit from a cross-fertilisation between a number of subcommittees in joint initiatives, meetings and activities.
When joining us you will have easy access to consolidated information collected from scattered sources.

How to participate?

All information will be made easily accessible by a number of electronic services.

Membership is for free.

Visit our Web-Site: http://www.isst.fhg.de

Statute of EASST

Name

European Association of Software Science and Technology

Location

The Association is located in Berlin/Germany.

Legal Status

The Association is a non-profit organization under German law (»gemeinnütziger eingetragener Verein«).

Purpose and Nature of Activities

The purpose of EASST is to promote the development of science and engineering on software intensive-systems, that play an increasing role in Europe´s way into the information society. It therefore supports education and qualification in software science and engineering, advices decision makers on appropriate measures, and informs the general public on the impact of technology developments.

The Association will

1. organize the exchange of information and spread research results by appropriate means to the community
2. provide help in the coordination of initiatives and projects in the area
3. organize and/or sponsor conferences like ETAPS and other professional meetings
4. coordinate its activities with other professional associations with the goal to give birth to a joint European association in informatics.

Membership

Ordinary membership in the association is open to individuals and legal entities, including other professional associations that support the goals of the EASST.
Associated membership may be obtained by members of other professional societies after proper agreement between them and EASST. Membership applications are requested in written form as determined by the board.

Membership Fee

A membership fee is not collected initially and may be collected later on, only after a decision taken by the membership at large.

Termination of Membership

Membership may be terminated by the member's resignation.
Membership will be terminated if the interest of the member in the membership in EASST vanishes. Indication of lost interest is abstention from decisions taken in EASST in electronic ballots for more than four times consecutively.
Membership will also be terminated if a membership fee due according to decision taken by the membership at large is not paid after its invoicing and after a second request.

Organs and Officers

General Assembly
The membership at large constitutes the general assembly of EASST. The General Assembly elects members of the board of EASST once every two years.
The General Assembly meets at least once a year to receive the annual report of the board including a financial and an activities report. An acceptance vote is expected four weeks after the issue of the report.
The General Assembly votes on the statutes of EASST not later than one year after its constitution, and on further amendments to the statute as well as on the dissolution of the association.

Board

The Board consists of the president, the vice president, the treasurer, the secretary, and four other board members without a particular portfolio.

Voting

Voting takes place in written form as determined by the board. The acceptance/rejection of the statutes, the amendment of the statute and the dissolution of the association require a two thirds majority of the members taking part in the vote.

Termination

In the event of the dissolution of the Association any remaining fund shall be disposed of in a manner determined by the General Assembly so as to support the purposes of EASST.

Application Form

I wish to become a member of EASST.

Please complete the following:

Name, First Name_____

Title_____

Company/University_____

Position_____

Street_____

Postal Code, City_____

Phone_____

Fax_____

E-Mail_____

Date and Signature_____

and return this form as soon as possible to:

EASST c/o
Herbert Weber
Fraunhofer-Institut für Software- und Systemtechnik
Mollstraße 1
D-10178 Berlin

Fax: +49 (0) 30/2 43 06-1 99
E-Mail: herbert.weber@isst.fhg.de