

THE EASST NEWSLETTER

Volume 13

June 2006



European Association of Software Science and Technology



EASST Board:

Prof. Dr. Tiziana Margaria (President),
email : tiziana.margaria@cs.uni-goettingen.de

Prof. Dr. Hartmut Ehrig (Vice-President),
email : ehrig@cs.tu-berlin.de

Dr. Michael G. Hinchey (Treasurer),
email : Michael.G.Hinchey@nasa.gov

Dr. Julia Padberg,
email : padberg@cs.tu-berlin.de

Prof. Dr. Marie-Claude Gaudel,
email : Marie-Claude.Gaudel@lri.fr

Dr. Maura Cerioli,
email : astes@disi.unige.it

Dr. Michel Wermelinger,
email : m.a.wermelinger@open.ac.uk

Dr. 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
Fakultät IV - Elektrotechnik und Informatik
Sekt. 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.....	J. Padberg	4
Summary of <i>GPSL: A Programming Language for Service Implementation...</i>	D. Cooney, M. Dumas, P.Roe	5
The SOFTWARE ANALYSIS AND VERIFICATION Column.....	J. Knoop	10
The SOFTWARE ARCHTECTURE Column The CommUnity Workbench.....	C. Oliveira, M. Wermelinger	14
The VISUAL LANGUAGES Column Tool Integration by Model Transformations based on the Eclipse Modeling Framework...	K. Ehrig, and G. Taentzer. D. Varró	25
The Minutes of the General Assembly	T. Margaria	37
Reports of the Extended Board		39
Report on ETAPs	J. Knoop	42
Report on FASE	R. Heckel	48
Report on ACCAT	H. Ehrig	50
Report on FESCA	R. Reussner	52
EASST Flyer		60
Application Form		63



Welcome

Dear EASST Members,

again I hope to present an interesting and worthwhile Newsletter. First I want to congratulate the winners of the EASST-award at ETAPS 2006, namely Dominic Cooney, Marlon Dumas and Paul Roe with their paper *GPSL: A Programming Language for Service Implementation*. In this volume we present a short summary of the award-winning paper. And there are interesting contributions in the columns, the minutes of the General Assembly and reports on ETAPS, FASE, ACCAT and FESCA.

As a result of the growing size of the EASST-Newsletter we have refuelled the initiative of an open access journal for EASST. The newsletter will from now on be more like a “*real newsletter*” concentrating on facts and issues of the scientific life. In contrast to that the new journal **Electronic Communications of EASST** (short EC-EASST) presents original contributions and is a fully refereed journal that provides a forum for practitioners, educators and researchers for disseminating innovative research in the area of software and system technology. The journal is also intended as a medium which rapidly publishes conference and workshop proceedings, provided that all contributions are original and peer reviewed. EC-EASST is an open access on-line journal, i.e. it provides unlimited access to all contributions which are published electronically only. In this way, EC-EASST provides the full-text access to all papers and supports the provision of fast and broad feedback to published work. The first volumes of EC-EASST will be published online in autumn 2006 and will cover several of the satellite events of 3. Int. Conference on Graph Transformation. You will be informed via the EASST-mailing list.

Yours sincerely,

Julia Padberg
(EASST-Secretary)

PS.: Please send any contribution to me or one of the column editors, if you would like to have it in the forthcoming newsletter next December.



GPSL: A Programming Language for Service Implementation (Extended Abstract)

Dominic Cooney* ** and Marlon Dumas* and Paul Roe*

*Queensland University of Technology, Australia

**Microsoft Corporation

1 Introduction

There is an increasing acceptance of Service-Oriented Architectures as a paradigm for software application integration. In this paradigm, independently developed and operated applications are exposed as (web) services that are then interconnected using standard protocols and languages [WCL⁺05]. While the technology for developing basic services and interconnecting them on a point-to-point basis has attained some maturity, there remain open challenges when it comes to implementing service interactions that go beyond simple sequences of requests and responses or that involve many participants.

A number of recent and ongoing initiatives aim at tackling these challenges. These initiatives can be classified into conservative extensions to mainstream programming languages and novel service-oriented programming languages. The former provide metadata-based extensions for web service development on top of object-oriented programming languages. For example Microsoft Web Services Extensions, Windows Communication Foundation, Apache Axis and JSR-181, can be placed in this category. While these extensions are suitable for dealing with bilateral interactions and simple forms of concurrency and correlation, capturing complex interactions with these libraries remains daunting. On the other hand, a number of service-oriented languages have been proposed, ranging from research proposals (e.g. XL [FGK02, FGK03]) down to standardisation initiatives, most notably the Business Process Execution Language for Web Services (BPEL) [ACD⁺03].

BPEL facilitates the development of services that engage in concurrent interactions and incorporates a declarative correlation mechanism, thus addressing some limitations of bespoke conservative language extensions. Nonetheless, it fails to provide direct support for typical service interaction scenarios. In [BDH05], a number of patterns of service interaction are proposed. It is shown that while BPEL directly supports the most basic of these patterns, it fails to address the needs of more complex scenarios. In particular, BPEL has problems dealing with one-to-many interaction scenarios with partial synchronisation.

The analysis of BPEL in [BDH05] suggests that web service implementation requires novel programming abstractions for dealing with advanced forms of concurrency, synchronisation, and message correlation. Accordingly, this paper presents a programming language, Gardens Point Service Language (GPSL), that integrates: (i) **dedicated messaging constructs** both for interacting with the other services

via SOAP and for structuring the internal implementation of services; (ii) **XQuery** [BCF⁺05] **expressions** integrated with imperative constructs; (iii) concurrent messaging based on **join-calculus** [FG96]; and an approach to **message correlation** based on message filters. For a more detailed introduction to GPSL the reader is referred to the full version of this extended abstract, which was presented at ETAPS'2006 [CDR06]. A compiler implementation of GPSL is available online¹.

2 Overview of GPSL

To illustrate the basic features of GPSL, we consider the implementation of a simple 'echo' service:

```
declare interface Echo {
  declare operation Shout in action = 'urn:echo:shout' out
}
declare service EchoService implements Echo {
  Shout($doc, Reply) { Reply($doc) }
}
```

GPSL has explicit contract and service declaration elements. Metadata from contracts are used by the compiler to provide types to operations. For example, the *Echo* contract has one operation, *Shout*. *Shout* is declared as an *in-out* operation and by convention in GPSL has two parameters: one for data, and the other for a channel to send the reply on. When a *Shout* operation message is received, in the case of *EchoService* the first parameter is bound to the body of the SOAP envelope and the second parameter is bound to the WS-Addressing (WS-A) reply-to SOAP header.

EchoService declares *implements Echo* and includes a block guarded by a label *Shout* that takes two parameters. The *Shout* label refers to an operation in the *Echo* contract, so whenever the service receives a message with SOAP action *urn:echo:shout* the service executes the corresponding block of code.

The *\$doc* parameter is bound to the body of the SOAP message and, by convention, the *Reply* parameter is bound to the WS-A reply-to header. *Reply* is opaque and the capability can only be passed to another service or exercised to send a message. The syntax for sending a message is to write the channel variable and a parameter list in parentheses. In this example, *EchoService* sends in the reply the data it received in the request.

The data model of GPSL has two kinds of values: XML data, such as the element *Say*, and channels, such as *Reply*. All XML expressions in GPSL are XQuery expressions. For example, *element Say* is an example of the XQuery computed element constructor. This ability to construct new XML data distinguishes XQuery from the less powerful XPath. However XQuery alone is not sufficient for implementing services because it is a pure functional language with no messaging constructs. Moreover, there are some semantic tensions between XQuery's flexible evaluation semantics and messaging, because it is difficult to determine when a message will be sent or received. To avoid these tensions, GPSL is based on a stratified approach in which imperative constructs are used for messaging whereas XQuery is used for expressions.

Messaging in GPSL is asynchronous; this encourages programmers to write services that make concurrent requests rather than sequences of request/responses, although this RPC programming style is also

¹<http://www.serviceorientation.com>



possible in GPSL. GPSL's means of spawning concurrent threads derives from asynchronous messaging: Services may also have private labels, and can start a concurrent block of code by sending an internal message to a private label.

Synchronisation is achieved through blocks of code guarded by multiple labels. Such multi-label guards are called *concurrency patterns* and are inspired by the join calculus. A block of code guarded by a concurrency pattern is executed when messages are available on all labels. For example, in the following code snippet, local messages *ResultA* and *ResultB* are sent in two different blocks of code *A()* and *B()* which we assume are executed concurrently (although their spawning is not shown). When both messages are available, then the rule at the bottom is reduced and the corresponding block of code is executed.

```
A() {
  ...
  ResultA(...) (: produce message ResultA :)
}
B() {
  ...
  ResultB(...) (: produce message ResultB :)
}
ResultA($a) & ResultB($b) (: this is a join pattern :) {
  (: executed when ResultA and ResultB are available :)
}
```

3 A Comparative Example: One-to-Many Send-Receive

We consider an interaction pattern where a service sends messages and collects responses before continuing. In this example we implement a broker service that solicits bids from a set of bidders, and collects responses, keeping track of the best (in this example, lowest) bid received. Bids are collected until a time-out occurs.

```
declare service Broker {
  InitiateAuction($env) {
    (: solicit bids :)
    for $bidder in $env/Bidders do
      $bidder: SolicitBid($env/Item, Reply)
    done;
    OutstandingBids(util:length($env/Bidders));

    (: start timer :)
    let $timeout := 'soap.inproc://timer' in
    $timeout: Time(10000, TimedOut);

    NoBids()
  }
  OutstandingBids($n) & Reply($bid) & NoBids() {
    Winning($bid);
    Decrement($n)
  }
  OutstandingBids($n) & Reply($bid) & Winning($best) {
    Winning(if xs:decimal($bid/Amount) < xs:decimal($best/Amount) then $bid else $best);
  }
}
```

```
    Decrement($n)
  }
  Decrement($n) {
    let $n := xs:int($n) - 1 in
    if xs:int($n) = 0 then
      BiddingFinished()
    else
      OutstandingBids($n)
    end
  }
  BiddingFinished() & Winning($bid) { (: process winning bid :) }
  TimedOut() & Winning($bid)      { (: fault or process winning bid :) }
  TimedOut() & NoBids()           { (: fault :) }
}
```

This program sends n *SolicitBid* messages. Although the *for* loop is sequential, the *SolicitBid* messages are sent in a non-blocking manner. The first bid received consumes the *NoBids* message and becomes the winning bid. Subsequent bids are compared to the winning bid. Messages *OutstandingBids* and *WinningBid* are used to capture state. They carry data for the number of outstanding bids and the best bid received. It is possible to check that this service correctly treats concurrent bids because contention for the *WinningBid* message acts as a mutual exclusion.

Coding the above scenario in BPEL is complicated by several factors. First, given that the set of partners to which bid requests are sent is not known in advance, dynamic addressing is required. In BPEL, this requires manual assignment of endpoint references to *partner links*. Second, BPEL lacks high-level constructs for manipulating collections. Thus, capturing this scenario requires the use of *while* loops and additional book-keeping. Third, there is no direct support in BPEL for interrupting the execution of a block when a given event (e.g. a timeout) occurs. To achieve this, it is necessary to combine an event handler with a fault handler, such that the event handler raises a fault when the nominated event occurs and the fault handler catches this artificially created fault. This causes the immediately enclosing scope to be stopped. In GPSL, such interruption can be achieved simply by adding a join pattern that matches the event in question (in this case, the timeout). The interested reader will find the full BPEL implementation of a similar scenario in the code repository of the service interaction patterns site.² This implementation comprises around 150 lines of BPEL code excluding comments, partly due to the verbosity of the XML syntax, but also because of the more fundamental drawbacks of BPEL mentioned above.

4 Conclusion

We have presented examples illustrating the GPSL language and its suitability for implementing common patterns of service interaction. GPSL cohesively integrates SOAP messaging, XQuery expressions, and join-calculus-based declarative concurrency extended with a message filtering mechanism that can be used to capture various correlation scenarios. Examples of the cohesive integration between messaging, XML data manipulation and concurrency in GPSL include: (i) the interplay between sending messages and spawning concurrent threads on the one hand, and receiving messages and synchronising threads

²See code sample “One-to-many send/receive with dynamically determined partners” at <http://www.serviceinteraction.com>

on the other; (ii) description of message recipients through dynamically constructed XML data; (iii) concurrency patterns describing thread-safe access to XML data; and (iv) consistent treatment of inter- and intra-service messages.

GPSL could be extended to address other difficult aspects of service implementation such as transactions and faults. We expect to address these areas by leveraging the messaging and concurrency features, for example, by surfacing faults as messages. We also plan to introduce a garbage collection technique to reclaim resources when it is detected that a given message will not be consumed.

Acknowledgments. The first author completed this work while working at Microsoft. The second author is funded by a fellowship co-sponsored by Queensland Government and SAP.

References

- [ACD⁺03] T. Andrews, P. Curbera, H. Dholakia, Y. Golland, 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, 2003.
- [BCF⁺05] Scott Boag, Don Chamberlin, Mary F. Fernández, Daniela Florescu, Jonathon Robie, and Jérôme Siméon. XQuery 1.0: An XML query language. W3C Working Draft, April 2005.
- [BDH05] A. Barros, M. Dumas, and A.H.M. ter Hofstede. Service interaction patterns. In *Proceedings of the 3rd International Conference on Business Process Management*, Nancy, France, September 2005. Springer Verlag. Extended version available at: <http://www.serviceinteraction.com>.
- [CDR06] Dominic Cooney, Marlon Dumas, and Paul Roe. GPSL: A programming language for service implementation. In *Proceedings of the 8th International Conference on Fundamental Approaches to Software Engineering (FASE)*, Vienna, Austria, March 2006. Springer Verlag.
- [FG96] Cédric Fournet and Georges Gonthier. The reflexive chemical abstract machine and the join calculus. In *The 23rd ACM Symposium on Principles of Programming Languages (POPL)*, pages 372–385, January 1996.
- [FGK02] Daniela Florescu, Andreas Grünhagen, and Donald Kossmann. XL: an XML programming language for Web service specification and composition. In *International World Wide Web Conference*, Honolulu, HI, USA, May 2002.
- [FGK03] Daniela Florescu, Andreas Grünhagen, and Donald Kossmann. XL: A platform for Web services. In *Conference on Innovative Data Systems Research (CIDR)*, Asilomar, CA, USA, January 2003.
- [WCL⁺05] S. Weerawarana, F. Curbera, F. Leymann, T. Story, and D.F. Ferguson. *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable-Messaging, and More*. Prentice Hall, 2005.



The “Software Analysis and Verification” Column

Jens Knoop *

*Institute of Computer Languages, Vienna University of Technology

Editorial

In this issue of the “Software Analysis and Verification” Column I would like to draw your attention to the two major events, which are sponsored by EASST this year. These are the *9th European Joint Conferences on Theory and Practice of Software, ETAPS 2006*, which recently took place in Vienna, Austria, and the forthcoming *2nd International Symposium on Leveraging Applications of Formal Methods, Verification, and Validation, ISoLA 2006*, which is going to take place in mid November this year on Cyprus.

The first of these two EASST sponsored events, ETAPS 2006, took place from 25th March to 2nd April 2006 in Vienna, Austria. It was hosted by the Vienna University of Technology, and organized by the Compilers and Languages Group of the Institute of Computer Languages. In addition to the 5 member conferences, CC, ESOP, FASE, FoSSaCS, and TACAS, ETAPS 2006 featured 18 satellite events and 2 tutorials. Together, these events attracted more than 700 participants from all over the world which again gave evidence for the fact that ETAPS is indisputably the primary European forum for academic and industrial researchers working on topics relating to software science. A detailed report on ETAPS 2006 can be found in this issue of the EASST Newsletter, photos taken at the conference, especially at its social events, can be found at the conference homepage at www.complang.tuwien.ac.at/etaps06.

The second major event, which EASST sponsors this year, is the forthcoming 2nd edition of ISoLA. Following the successful first edition of ISoLA in the fall of 2004, this second edition of ISoLA will feature in addition to its main programme an array of special tracks and special sessions on selected hot topics. Some of these special tracks and sessions match the scope of this column and are of particular interest for its readers. It is worth noting that the call for papers for ISoLA 2006 and its special tracks and sessions is still open. Readers of this column are invited to inspect the information on these tracks and sessions at ISoLA 2006 in the conference calendar below or to visit the homepage of ISoLA 2006 at www.seds.informatik.uni-goettingen.de/isola/ for full details. I look already forward to seeing you at this or another forthcoming EASST sponsored event.

Last but not least, I do not want to conclude this issue of the column without inviting you again to contribute to it. Possible topics include but are not limited to conference and project announcements and reports, book reviews, position statements as well as original contributions. I look forward to your contributions, which shall directly be sent to the editor at knoop@complang.tuwien.ac.at.

Have a nice summer time!



.....

Conference and Summer School Announcements

Call for Papers

- 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2006), 26-29 September 2006, Timișoara, Romania.
synasc06.info.uvt.ro/index.php
Submission Deadline: 5 June 2006
- International Workshop on Software Certification (CERTSOFT 2006), 26-27 August 2006, McMaster University, Hamilton, Ontario, Canada. In conjunction with FM 2006.
fm06.mcmaster.ca/certsoft
Submission Deadline: 9 June 2006
- Workshop on Constraints in Software Testing, Verification and Analysis (CSTVA 2006), 25-29 September 2006, Nantes, France. Co-located with CP 2006.
www.irisa.fr/manifestations/2006/CSTVA06/index.htm
Submission Deadline: 23 June 2006
- 2nd Annual Haifa Verification Conference, 23-26 October 2006, Haifa, Israel.
www.haifa.il.ibm.com/Workshops/verification2006/
Submission Deadline: 30 June 2006
- 2nd International Symposium on Leveraging Applications of Formal Methods, Verification, and Validation (ISoLA 2006), 15-19 November 2006, Cyprus.
www.seds.informatik.uni-goettingen.de/isola/
Submission Deadline: 15 June 2006
 - 2nd International Workshop on Automated Specification and Verification of Web Systems (WWV 2006). Track of ISoLA 2006.
www.dsic.upv.es/workshops/wwv06
Submission Deadline: 3 July 2006
 - Formal Aspects of Validating and Verifying Very Large Systems. Track of ISoLA 2006.
 - Formal Approaches to the Specification and Verification of Sensor Networks. Thematic Session at ISoLA 2006.
- 1st Asian Working Conference on Verified Software (AWCVS 2006), 29 - 31 October 2006, Macao SAR, China. Satellite event of ICFEM 2006.
www.iist.unu.edu/www/workshop/AWCVS2006/
Submission Deadline: 10 July 2006
- 8th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI 2007), 14-16 January 2007, Nice, France.



research.microsoft.com/vmcai07
Submission Deadline: 8 September 2006

- 5th Annual IEEE CS TC-uARCH/ACM SIGMICRO/SIGPLAN International Symposium on Code Generation and Optimization (CGO 2007), San Jose, California, March 2007.
www.cgo.org/cgo2007/
Submission Deadline: 8 September 2006

Call for Participation

- 26-30 June 2006
2nd TAROT Summer School on Testing of Software and Communicating Systems, Toledo, Spain.
www.info-ab.uclm.es/tarot/
- 4 July 2006
Program Analysis for Security and Safety Workshop Discussion (PASSWORD 2006), Nantes, France. Co-located with ECOOP 2006.
research.ihost.com/password/
- 17-20 July 2006
International Symposium on Software Testing and Analysis (ISSTA 2006), Portland, Maine.
www.cis.udel.edu/issta06/
- 26-28 July 2006
Design, Specification and Verification of Interactive Systems (DSV-IS 2006), Trinity College, Dublin, Ireland.
www.dsvvis2006.org
- 15-16 August 2006
3rd International Verification Workshop (VERIFY 2006), Seattle, Washington. In connection with IJCAR 2006 at FLoC 2006.
www.ags.uni-sb.de/~omega/workshops/Verify06/
- 16 August 2006
Workshop on Disproving, Non-Theorems, Non-Validity, Non-Provability (DISPROVING 2006), Seattle, Washington. In connection with IJCAR 2006 at FLoC 2006.
www.cs.chalmers.se/~ahrendt/FLoC06-ws-disproving/
- 21 August 2006
International Workshop on Software Verification and Validation (SVV 2006), Seattle, Washington. In conjunction with FLoC 2006.
www.comp.nus.edu.sg/~abhik/svv06/
- 21 August 2006
1st Workshop on Verification and Debugging, Seattle, USA. Associated with CAV 2006 at FLoC



2006.

www.ist.tugraz.at/vandd.html

- 31 August 2006
2nd Workshop on Graph Transformation for Verification and Concurrency (GT-VC 2006), Bonn, Germany. Satellite workshop of CONCUR 2006.
depend.cs.uni-sb.de/concur2006/
- 16 - 20 September 2006
15th IEEE CS TCPP/TCAA / ACM SIGARCH / IFIP WG 10.3 International Conference on Parallel Architectures and Compilation Techniques (PACT 2006), Seattle, Washington.
www.cs.virginia.edu/~pact2006/
- 18-19 September 2006
6th International Workshop on Automated Verification of Critical Systems (AVoCS 2006), Nancy, France.
avocs06.loria.fr/
- 27-29 September 2006
6th Sixth IEEE International Workshop on Source Code Analysis and Manipulation (SCAM 2006), Philadelphia, Pennsylvania.
www.ieee-scam.org/
- 23-26 October 2006
4th International Symposium on Automated Technology for Verification and Analysis (ATVA 2006), Beijing, China.
lcs.ios.ac.cn/~atva06/



The CommUnity Workbench

Cristóvão Oliveira * and Michel Wermelinger **

*Department of Computer Science, University of Leicester
University Road, Leicester LE1 7RH, UK
co49@mcs.le.ac.uk

**Computing Department, The Open University
Walton Hall, Milton Keynes MK7 6AA, UK
m.a.wermelinger@open.ac.uk

1. Introduction

COMMUNITY is a parallel program design language initially developed to show how programs fit into Goguen's categorical approach to General Systems Theory. Since then, the language and its framework have been extended to provide a formal platform for architectural design of open, reactive, reconfigurable systems.

The COMMUNITY Workbench implements directly over Java the constructions of coordination and distribution that give semantics to the COMMUNITY approach as an architectural description language. This paper is an extension of a formal demo presented at ICSE'04 [6]. The architectural approach, implemented in the current version of the tool, describes component types as COMMUNITY designs, interactions between component instances as simple connections, or with architectural connectors as described in [3]. The COMMUNITY language takes into account the properties of the "physical" distribution topology of locations and communication links, reflecting distribution and mobility at the computational level (design level).

The running example is a client-server system with very generic components and connectors. The client is a mobile laptop, and uses a mobile printer as the server. The communication is asynchronous and is implemented through a message passing connector with a buffer. The laptop does not control its own location, whereas the printer, whenever it is not co-located with the laptop, interrupts the reception of messages and tries to move to the laptop's position. This is achieved with a distribution connector that implements a "follow me" policy. The example is a mixture of some examples given in [5]. A more substantial example can be found in [7] where we used COMMUNITY to model the GSM handover protocol that takes place when a mobile phone moves from the scope of one antenna to another one. Additionally, the COMMUNITY website [8] provides an updated and general account of the various research strands.

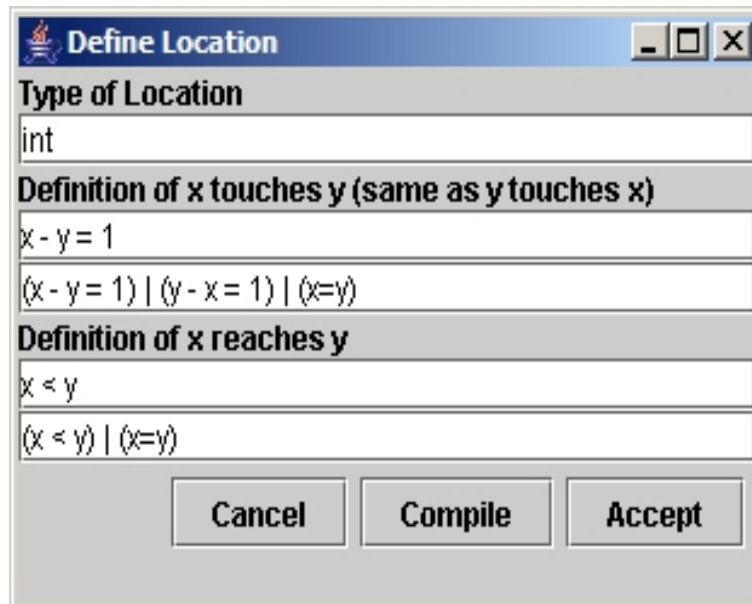


Figure 1. The location editor.

2. CommUnity

COMMUNITY designs are in the style of UNITY programs [1], but they also combine elements from IP [4]. However, COMMUNITY has a richer coordination model and, even more important, it requires interaction between components to be made explicit. In this way, the coordination aspects of a system can be separated from the computational aspects and externalized, making explicit the gross configuration of the system in terms of its components and its interactions. Each configuration can be transformed into a single, semantically equivalent, design that represents the whole system. Moreover architectures are not just depicted through lines and boxes; they are diagrams in the sense of category theory [2].

In order to model systems that are location-aware, COMMUNITY adopts an explicit representation of the space within which movement takes place, but does not assume any specific notion of space. So, COMMUNITY separates explicitly three dimensions: Computation, Coordination, and Distribution/Mobility. The remaining of this section explains how the separation of the three dimensions is achieved in COMMUNITY.

2.1. Components

A COMMUNITY design consists of a set of location variables, a set of typed channels and a set of actions. COMMUNITY is independent of the actual data types used, but the Workbench provides a fixed set of types: integer and real numbers, booleans, lists, arrays, records, and enumerations. There are *input*, *output* and *private* channels. Input channels are read-only. Output and private channels are called *local* channels and cannot be changed by the environment. A design with input channels is *open* in the sense

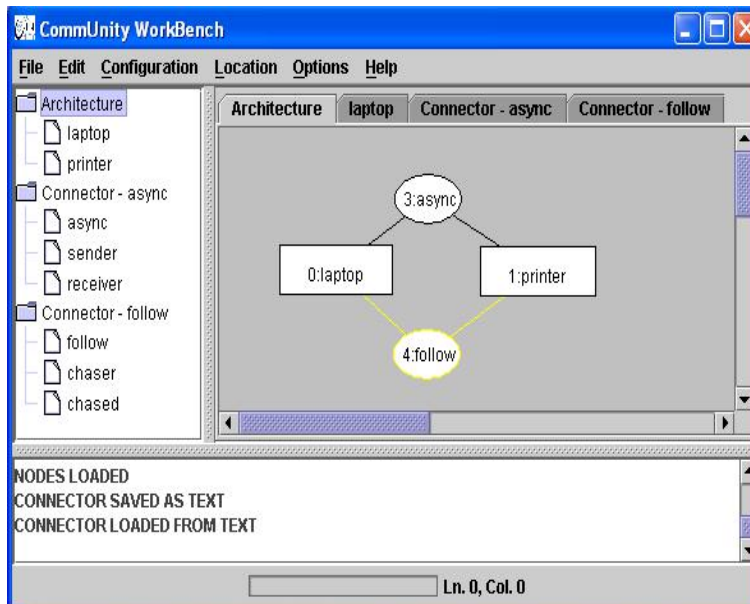


Figure 2. The main window.

that it needs to be connected to other components of the system to read data, as explained further on.

Location variables can be declared as input or output in the same way as channels but are all typed with a special sort – Location. Input locations are read from the environment and cannot be modified by the component. In the designs that are location-aware, it is explicit how their constituents are mapped to the positions of the fixed space. This corresponds to the distribution aspect. Each local channel is associated with a location variable. Each action name is associated with a set of a locations, meaning that the execution of the action is distributed over those locations. A modification in the value of a location variable entails the movement of the channels and actions located there. The mobility aspect is made explicit here. Hence, if “l” is an input location variable, the movement of any constituent located at “l” is under control of the environment. Output locations can only be modified locally.

There are *private* and *shared* actions. Each action has a name and a set of distributed bodies. Each body has two guards, and a set of deterministic assignments. The *safety* and *progress* guards are propositions over local channels—because input channels are not under control of the design—and establish an interval in which the enabling condition of the action must lie, the safety guard being the lower bound. When the guards are equivalent, we write only one of them. At each execution step, one of the actions whose enabling condition holds of the current state is selected, and its assignments are executed atomically in parallel. We write `skip` to denote the absence of assignments. A design is organized to be a unit of execution and defines the computation aspect.

The reactive designs to be used in our architecture are a laptop, which simulates the production of documents in “ps” or “pdf” format and sends them to a printer, which receives and prints documents in the same format.



```
design laptop
inloc ll
out outfile@ll : enum(ps, pdf)
prv saved@ll : bool
do edit@ll : true, false -> saved := false
[] save_ps@ll : ~saved -> outfile := ps || saved := true
[] save_pdf@ll : ~saved -> outfile := pdf || saved := true
[] send@ll : saved -> skip

design printer
inloc lp
in infile : enum(ps, pdf)
prv busy@lp : bool;
    printfile@lp : enum(ps, pdf)
do get@lp : ~busy -> printfile := infile || busy := true
[] prv print@lp : busy -> busy := false
```

The design “laptop” has an input location variable “ll” (location of laptop), an output variable “outfile” which holds a Postscript or PDF file, and a private boolean variable stating whether the file is saved. The “laptop” is centralised, in the sense that all its data and code are located in the same place. For instance both local variables and the actions are mapped to “ll”. The design “printer” is also centralized in the location represented by “lp”. Notice that the locations of the laptop and printer are given by input location variables, as neither of them controls its own position. In both designs, “laptop” and “printer”, every action has a single body, and the order of execution is constrained by the use of state variables (“saved” and “busy”). All actions are public (i.e., may be coordinated with others) except action “print”. Notice also that “printer” makes a local copy of the received file to “printfile” because input variables are controlled by the environment and may therefore change their value anytime.

2.2. Connectors

In COMMUNITY, the model of interaction between components is based on shared action synchronization and the sharing of input channels of a component with output channels of other components. The sharing of the input location variables of a component with output location variables of other components specifies the distribution over the space. Notice that private actions and private channels are not involved in interactions. Moreover, it is not allowed to (in)directly share two output channels, or synchronize two actions of the same node. COMMUNITY, unlike other program design languages, requires interaction between components (name bindings) to be made explicit.

In the current version of the tool the interaction between components can be made using architectural connectors, which encapsulate behaviour of the interaction. The interaction between components made by simple connections or by architectural connectors corresponds to the coordination dimension. A connector is defined by a glue specification which is a design, and a set of roles which are also designs in COMMUNITY. The interaction between the glue and the roles are simple connections (a set of action

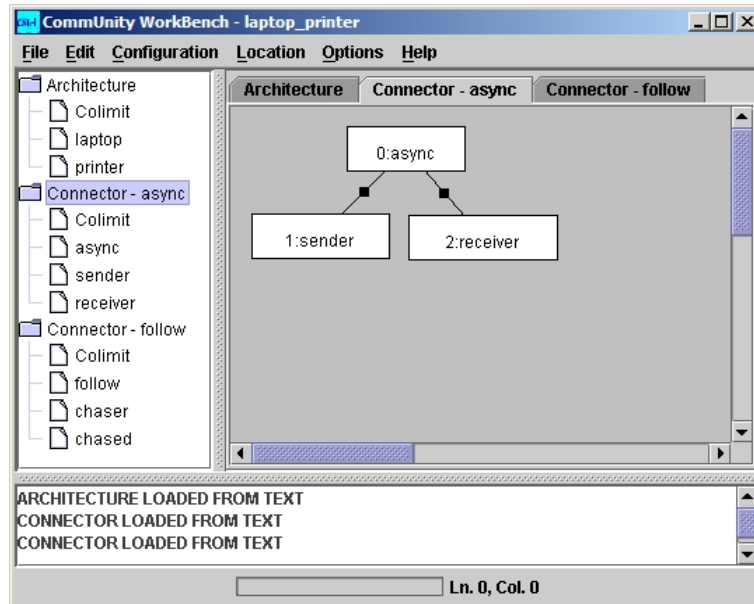


Figure 3. The connector “async”.

synchronizations and sharing of channels). The glue describes how the activities of the role instances are to be coordinated. The use of a connector in the construction of the architecture consists in the instantiation of its roles with specific designs. The instantiation of a role with a component is possible iff the component fulfils the requirements the role determines. Therefore, instantiation corresponds to a form of refinement.

In our architecture we use two connectors. The first is an asynchronous file passing connector with two roles, a sender and a receiver. The second is a distribution connector, and defines a pattern of mobility involving two components that have to be instances of, respectively, the roles “chased” and “chase”. In the mobility pattern thus defined, the instance of “chase” is moved to the location of the instance of “chased” whenever they are not co-located. In our example the laptop is “chased” and the printer “chase”s.

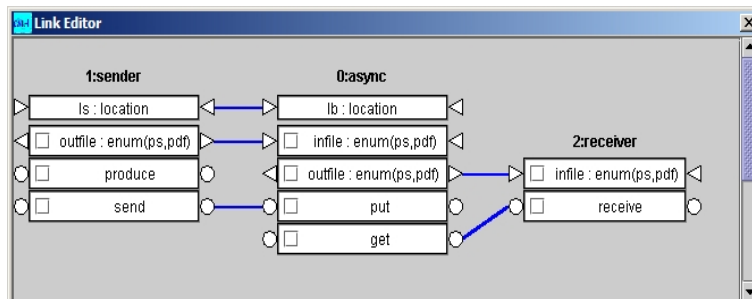


Figure 4. The glue-role bindings in connector “async”.

```

connector follow {
  glue
  design chase
    inloc lchased
    outloc lchaser
    do prv move@lchaser : lchaser /= lchased -> lchaser := lchased
  roles
  design chaser inloc l
  design chased inloc l
  configuration
    c:chase
    c1:chaser
    c2:chased
    attachments {c.lchased-c2.l c.lchaser-c1.l}
}

```

2.3. Configuration

A configuration has a very precise mathematical semantics, given by a diagram in a category whose objects are the designs and whose morphisms capture a notion of program superposition [1]. Any such categorical diagram can be transformed, by a universal categorical construction called *colimit*, into a single design that represents the whole distributed system. In our example the colimit obtained is a distributed component design which produces documents and prints them, and also includes the buffer. It's important to note that the colimit is calculated by the workbench; the user is not required to know category theory.

The relevant properties of the mobility space are captured by two binary relations over the domain. A relation “touches” means that two positions in the space are “in touch” with each other. Coordination among components takes place only when all the locations of all the actions involved in a synchronization are “in touch”. The “touches” relation is assumed to be reflexive and symmetric. The second binary relation “reaches” means that one position is “reachable” from the other. Movement of a component to

a new position is possible only when this position “is reachable” from the current one i.e. a location variable can take a new value if that value is of a location “reachable” from the current position represented by the current value of the location variable. Reachability is assumed to be a reflexive relation. In the COMMUNITY Workbench we show the two relations explicitly during a design execution as shown in Figure 8 – the white line represents the “touches” relation and the black directed line shows that the position of the source node reaches the position of the target node.

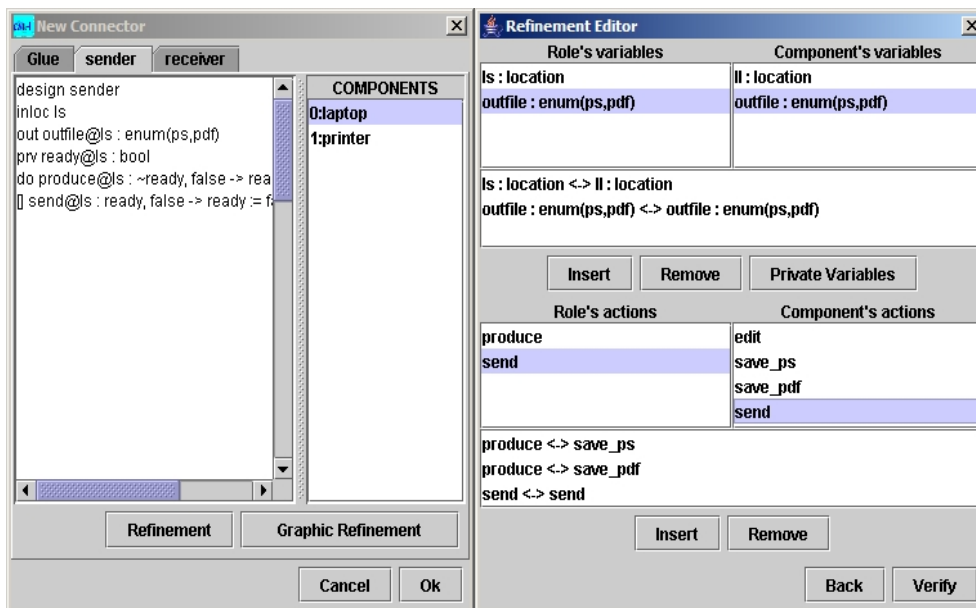


Figure 5. Refining sender with laptop.

3. Workbench

The Workbench requires a Java 1.4 runtime and is available as a self-installing application from the first author’s webpage. The installer was made for multiple platforms (Windows XP, Mac OS X, and Linux) with InstallAnywhere 5.5.1 Now [10]. Additionally, we used JavaCC 2.0 [9] to generate the COMMUNITY parser. The Workbench distribution includes some examples and a user manual.

The current version of the tool allows the user to:

- 1–Specify the location type and the relations “touches” and “reaches”;
- 2–Write COMMUNITY designs;
- 3– Design graphically architectural connectors;
- 4–Design graphically the architecture adding connectors to the current configuration and calculate its colimit;
- 5–Run a design (in particular the colimit of a configuration).

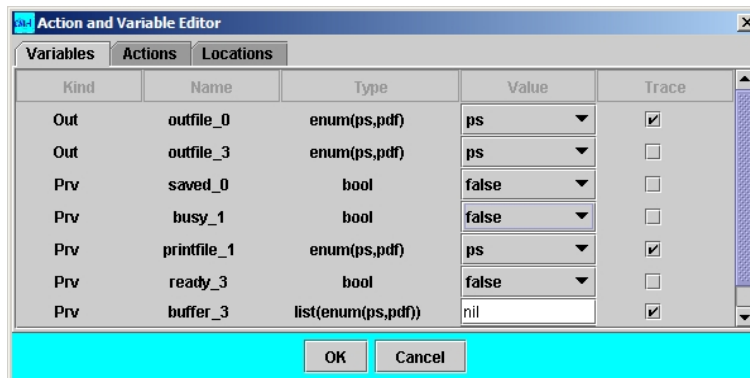


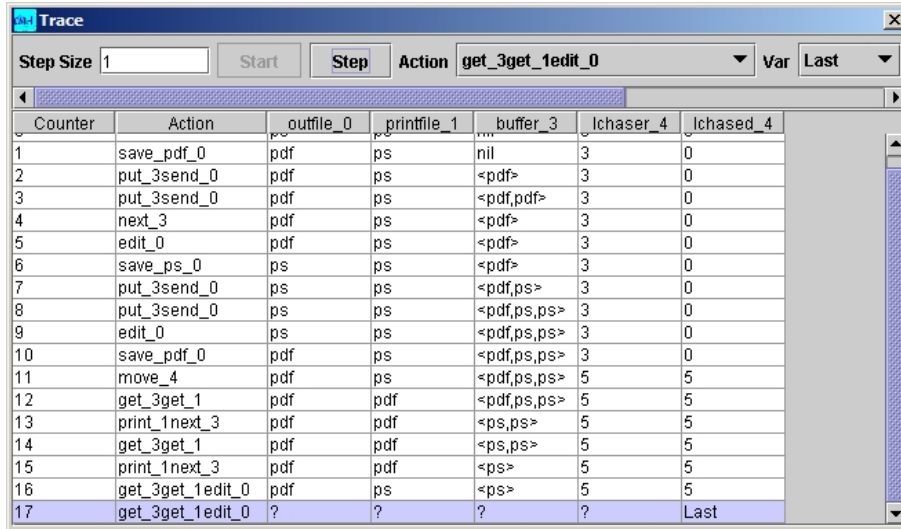
Figure 6. The action and variable editor.

1– The data type Location is defined in terms of the pre-defined types (integers, reals, etc...). In our example the locations are represented by integers. The “touches” and “reaches” relations are defined as boolean expressions with two parameters which are locations. In Figure 1 we show the definition of the data type Location and the expressions to implements the relations between locations. In our example, two locations are “touches” if they are consecutive numbers and one location is “reachable” from another if the first one is bigger than the last one. In Figure 1 the second line of the “touches” definition includes the reflexive and symmetrical closure, while the second line of “reaches” includes the reflexive closure.

2– As shown in Figure 2, the workbench has three panes. On the left are the trees of the architecture and the opened architectural connectors; each architecture has a list of component designs, together with one item for the resulting colimit design. On the right is one tabbed window for each item on the left, and below is the message area. In tabbed windows for designs, the user may write a new design or edit an existing one.

3– In the connector diagrams the user may do the same he can do in the architecture diagram (see step 4), but add and delete connectors. In the architecture (Figure 2) and the connector diagrams (Figure 3), each node is uniquely numbered and must be an instance of an existing design. The architectural shape for connector diagrams must be a star topology. The center of the star will be the glue of the connector.

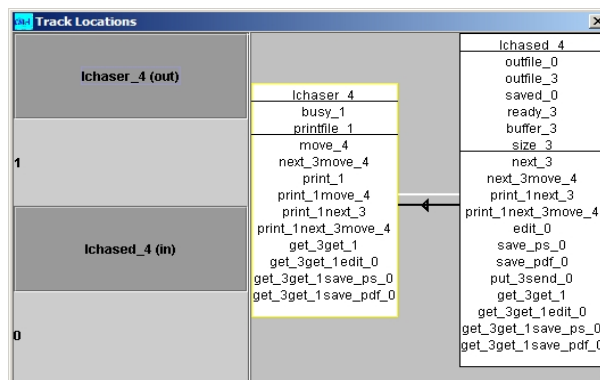
4– In the tabbed window for the architecture, the user may add and delete nodes, arcs and connectors, drag nodes around to change the diagram layout, double-click on an arc or select some nodes to invoke a graphic link editor (Figure 4) to see or set the bindings between the selected nodes. The editor presents to the user all the details of the selected nodes. The user can make direct connections between channels and locations and synchronizations between actions of different nodes. The workbench detects bindings that would (in)directly share output channels or output locations, or synchronize actions of the same node. A checked action or channel will be closed (i.e., become private) in the colimit, and automatically so will all actions synchronized (resp. channels shared) with it. In our example we show the details of the



Counter	Action	outfile_0	printfile_1	buffer_3	lchaser_4	lchased_4
1	save_pdf_0	pdf	ps	nil	3	0
2	put_3send_0	pdf	ps	<pdf>	3	0
3	put_3send_0	pdf	ps	<pdf, pdf>	3	0
4	next_3	pdf	ps	<pdf>	3	0
5	edit_0	pdf	ps	<pdf>	3	0
6	save_ps_0	ps	ps	<pdf>	3	0
7	put_3send_0	ps	ps	<pdf, ps>	3	0
8	put_3send_0	ps	ps	<pdf, ps, ps>	3	0
9	edit_0	ps	ps	<pdf, ps, ps>	3	0
10	save_pdf_0	pdf	ps	<pdf, ps, ps>	3	0
11	move_4	pdf	ps	<pdf, ps, ps>	5	5
12	get_3get_1	pdf	pdf	<pdf, ps, ps>	5	5
13	print_1next_3	pdf	pdf	<ps, ps>	5	5
14	get_3get_1	pdf	pdf	<ps, ps>	5	5
15	print_1next_3	pdf	pdf	<ps>	5	5
16	get_3get_1edit_0	pdf	ps	<ps>	5	5
17	get_3get_1edit_0	?	?	?	?	Last

Figure 7. The trace window.

opened architectural connector “async”. In the graphic link editor the circles are actions, inward pointing triangles are input channels, and outward pointing triangles are output channels. The user may add the created connectors to the main architecture. In our example we apply the two connectors “async” and “follow” (Figure 2) to the same components. The window to add connectors (Figure 5, left) has one pane for the glue and one for each role. In a role pane the user chooses on the right the component instance that will refine the current role and clicks the “Refinement” button. A dialog appears (Figure 5, right) where the user may map each channel/action of the role to the refined channel/action of the component. The workbench prevents wrong refinements, like refining an input channel by an output channel. If the role has no private actions or channels, then a graphic link editor can be used by clicking on the “Graphic Refinement” button.



Role	Channel/Action	Component Instance
lchaser_4 (out)	busy_1	lchaser_4
	printfile_1	lchaser_4
lchased_4 (in)	move_4	lchased_4
	next_3move_4	lchased_4
	print_1	lchased_4
	print_1move_4	lchased_4
	print_1next_3	lchased_4
	print_1next_3move_4	lchased_4
	edit_0	lchased_4
	save_ps_0	lchased_4
	save_pdf_0	lchased_4
	put_3send_0	lchased_4
	get_3get_1	lchased_4
	get_3get_1edit_0	lchased_4
	get_3get_1save_ps_0	lchased_4
	get_3get_1save_pdf_0	lchased_4

Figure 8. The group of collocated variables



There is a menu item to compute the colimit of a configuration, be it a single connector or a complete architecture. Being a design, the colimit can be opened in an editing window for inspection.

5– Before running a design, the user has to provide the initial values for all channels and location variables, and to choose which channels and actions should be traced (Figure 6). By default, numbers are initialized with zero, booleans with false, and only output channels and output location variables are traced.

The trace window Figure 7 gives great flexibility to test several scenarios. Shared actions can be selected for execution explicitly by the user or automatically by the tool, in fair or random mode. Private actions are always selected in a fair mode. There are also three choices for the value of any input channel that has not been connected to any output channel: it may be the same as in the previous execution step; it may be a random value; or it may be set by the user. During the trace the user can track values of the location variables as shown in Figure 8 and also see which channels and actions are located in the same position. In Figure 7 we show part of an execution of our example’s colimit.

4. Concluding remarks

The Workbench allows the separate definition of computation, coordination and distribution through distinct windows and dialogs, and the animation of the overall system. The architecture can be checked incrementally at various levels: linguistic errors in designs, invalid bindings and refinements in configurations, and unexpected behaviour during animation. For the latter, it is possible to probe specific scenarios by defining at each step the input values and the action to be executed. The environment also allows the separate saving of designs, connectors and architectures in a textual format that allows multiple users to reuse and adapt them for their needs.

The Workbench is part of an overall research effort into the formal foundations of various architectural concepts for coordinated mobile systems [8]. In particular, the Workbench provides a proof of concept for the stepwise design approach given at the start of section 3, by providing an integrated environment for architectural specification through box-and-line diagrams. However, a distinguishing feature of the COMMUNITY Workbench is the formal foundation it is based on, which provides a very precise semantics to those architectural diagrams.

5. Acknowledgments

We thank José Fiadeiro and Antónia Lopes for their support and feedback on the tool. We also thank João Feliciano, Helder Neto for implementing the initial version of the workbench, José Aires Camacho for implementing some new features and the output of the textual specification, Mário Costa for the graphics of the interactions between the components, and Brian Plüss and Frederico Palma for implementing the distribution and mobility aspects.

The development of this tool was supported by Fundação para a Ciência e Tecnologia through the PhD Scholarship SFRH/BD/6241/2001 of Cristóvão Oliveira and through project POSI/32717/00 (FAST—Formal Approach to Software Architecture), and by the European Commission through project IST-2001-32747 (AGILE—Architectures for Mobility) of the Global Computing Initiative.

References

- [1] K. M. Chandy and J. Misra. *Parallel Program Design—A Foundation*. Addison-Wesley, 1988.
- [2] J. L. Fiadeiro. *Categories for Software Engineering*. Springer-Verlag, 2004.
- [3] J. L. Fiadeiro, A. Lopes and M. Wermelinger. A Mathematical Semantics for Architectural Connectors. In *Generic Programming*, LNCS 2793, pp. 190-234. Springer-Verlag, 2003
- [4] N. Francez and I. Forman. *Interacting Processes*. Addison-Wesley, 1996.
- [5] A. Lopes, J. L. Fiadeiro and M. Wermelinger. Architectural Primitives for Distribution and Mobility. In *Proc. ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE 10)*, pp. 41-50, ACM Press, 2002
- [6] C. Oliveira and M. Wermelinger. The community workbench. In *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, pages 709–710. IEEE Computer Society, 2004.
- [7] C. Oliveira, M. Wermelinger, J. L. Fiadeiro, A. Lopes, Modelling the GSM handover protocol in CommUnity. *Proc. of the Workshop on Formal Foundations of Embedded Systems and Component-Based Software Architectures*, pages 3-25, Electr. Notes Theor. Comput. Sci. 141, no. 3, 2005.
- [8] The CommUnity website.
<http://www.fiadeiro.org/jose/CommUnity>
- [9] The Java compiler compiler.
<https://javacc.dev.java.net>
- [10] Zero G Software, InstallAnywhere.
<http://www.zerog.com>



Tool Integration by Model Transformations based on the Eclipse Modeling Framework

Karsten Ehrig* and Gabriele Taentzer** and Dániel Varró***

*University of Leicester, United Kingdom

**Technical University of Berlin, Germany

***Budapest University of Technology and Economics, Hungary

Abstract. *In the paper, we propose various approaches for tool integration based on model transformations over the Eclipse Modeling Framework (EMF). EMF is a key technology for tool integration, which provides a framework for developing domain-specific modeling languages by automatically generating Java code for model manipulation. Model transformations can be captured by graph transformation systems, which support visual specifications based on rules and patterns. Three levels of tool integration are identified: (i) model-level integration carries out model transformations in existing transformation tools by importing and exporting EMF models, (ii) interpreted EMF transformations take an EMF model of the transformation system, and manipulate EMF models according to the system by calling EMF interfaces, finally (iii) compiled transformer plug-ins generate stand-alone transformer programs in Java which are responsible for model manipulation.*

Keywords: model transformation, tool integration, graph transformation, Eclipse

1 Introduction

In model-driven software development, the Eclipse Modeling Framework (EMF) [EMF06] is becoming a key technology for tool integration. It is a framework for developing domain-specific modeling languages defined by an appropriate class diagram (metamodel) by automatically generating Java code which supports to create, modify, store, and load instances of the model. Moreover, it provides generators to support the basic editing of EMF models. EMF unifies three important technologies: Java, XML, and UML. Regardless of which one is used to define a model, an EMF model can be considered as the common representation that subsumes the others.

Graph transformation [EEPT06] provides a rule and pattern-based specification technique to manipulate graph-based models. Graph transformation approaches and tools frequently serve as an underlying framework for capturing model transformations with and between modeling languages at various levels of abstraction.

Model transformations have been identified as a key challenge in the SENSORIA European IST project.



The aim of SENSORIA is to develop a novel comprehensive approach to the engineering of software systems for service-oriented overlay computers where foundational theories, techniques and methods are fully integrated in a pragmatic software engineering approach. It focuses on global services that are context adaptive, personalisable, and may require hard and soft constraints on resources and performance, and plans to take into account the fact that services have to be deployed on different, possibly interoperating, global computers, to provide novel and reusable service-oriented overlay computers.

Model transformations aim to serve as a bridge between various SENSORIA tools used for the design, the discovery, the verification, the implementation or the deployment of services.

In the current paper, we propose to use EMF as general framework for integrating tools by model transformations captured by graph transformation techniques in order to integrate the advantages of both techniques. The envisaged tool integration can be carried out in three different ways:

1. **Model-level integration.** A first (traditional) approach (Sec. 3) is to use existing model transformation tools by providing appropriate facilities to import and export for EMF models. Starting from an EMF source model, it is first imported into the designated model transformation tool. Then the actual model transformation is carried out using the internal model representation of the transformation tool. Finally, the result of the transformation is exported into a target EMF model.
2. **Interpreted transformations of EMF models.** A second approach (Sec. 4) is to store model transformations also in the form of EMF models and write an interpreter, which takes a source model and transformation rules in its EMF representation as input, and generates the designated target model as output. In this solution, the transformation engine is a general one, which should be capable of transforming any models corresponding to EMF.
3. **Compiled transformer plugins in Java.** A third approach (Sec. 5) is to take a specification of a model transformation and compile it into a stand-alone transformer plugin (i.e. a Java program), which takes a source EMF instance as input and generates a target EMF instance as output.

As model transformation is becoming an engineering discipline (*transware* [VP04]), conceptual and tool support is needed for the entire life-cycle, i.e. the specification, design, execution, validation and maintenance of transformations. However, different phases of transformation design frequently set up conflicting requirements, and it is difficult to find the best compromise. For instance, the main driver in the execution phase is performance, therefore, a *compiled MT approach* (where a transformation is compiled directly into native source code) is advantageous. On the other hand, *interpreted MT approaches* (where transformations are available as models) have a clear advantage during the validation (e.g. by interactive simulation) or the maintenance phase due to their flexibility.

After a brief summary on EMF (2), Section 3 first details model-level integration techniques. Then Sec. 4 proposes the use of interpreted EMF transformations for tool integration. In Sec. 5, we outline how automatically generated stand-alone transformer plug-ins can serve as a transformation based integration technique. Finally, Secs. 6 and 7 conclude this paper.



2 Eclipse Modeling Framework (EMF)

The Eclipse Modeling Framework (EMF) [EMF06] provides a modeling and code generation framework for Eclipse applications based on structured data models. The modeling approach is similar to that of MOF, actually EMF supports Essential MOF (EMOF) as part of the OMG MOF 2.0 specification [EMO06]. The type information of sets of instance models is defined in a so-called core model corresponding to the metamodel in EMOF. The core (or metamodel for core models) is the Ecore model. It contains the model elements which are available for EMF core models in principle.

From an EMF model, a set of Java classes for the model and a basic, tree based editor can be generated. The generated classes provide basic support for *creating/deleting* model elements, persistency operations like *loading and saving*, and a *notification mechanism* to support the model-view-controller paradigm by sending events on model changes.

Relations between EMF model classes are handled by special EMF lists, extending the Java list classes. Moreover, EMF models can be used as underlying models in new application plugins. But in many cases, the EMF model by its own is not powerful enough to express the complete model behavior. Therefore the generated code can be extended by the developer in order to add new functionalities that are not expressed in the EMF model.

The practical relevance of EMF is clearly demonstrated by the fact that many software vendors (like IBM and Borland) have recently chosen to commonly use EMF models in their products to facilitate easy tool integration.

3 Model-Level Integration

The first approach (illustrated in Fig. 1) for integrating model transformations is to use the import / export functionalities of an existing model transformation tool like VIATRA2 [VIA06] and development tools for ATL [ATL06].

- In the first step, the designated source EMF model is imported into the model transformation tool either by generating some tool-specific model descriptions, or calling directly the model manipulation interfaces of the transformation tool from an EMF model instance.
- Then the entire model transformation is designed and executed using the designated model transformation tool. The result of the transformation is obtained in a tool-specific format, which is again either a textual or an in-memory representation of the target model.
- Finally, the target EMF model is generated from the tool-specific target model by calling the appropriate model manipulation methods of the target EMF API or generating an XMI 2.0 document (specific to the target modeling language) which can be parsed automatically by the EMF framework.

This approach primarily provides an off-line synchronization between EMF and tool-specific models, ²⁷i.e. the models are only integrated into EMF after the target model has been generated by the transformation. However, an advanced (on-the-fly) model-level integration approach may aim at the synchro-

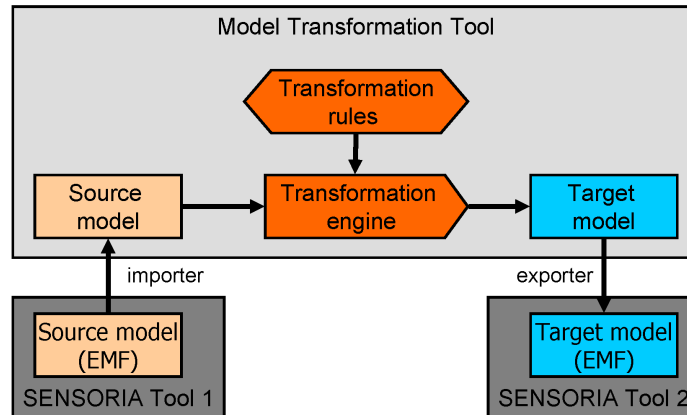


Figure 1: Model-level integration by existing model transformation tools

nization of notification mechanisms of EMF and the used model transformation framework. As a result, changes in both models are propagated immediately between the two platforms.

Consistency recovery. Since a model-level integration approach is basically independent from EMF, syntactic consistency of the generated target EMF models are typically checked only after the transformation (but not during the execution). Thus, in case of erroneous model transformations, errors are detected and recovery actions are performed late during the execution process. In any case the error checks depend on the functionalities of the used model transformation tool but most of these are typically post-transformation checks. The following approaches aim at providing an early (on-the-fly) error recovery by forcing transformations to use EMF interfaces for model manipulation. Furthermore, the transformation system has to be created within the model transformation tool, i.e. completely independent of EMF models.

4 Interpreted Transformations of EMF Models

The second approach (illustrated in Fig. 2) for performing model transformations is to take a specification of a model transformation and interpret the rules by an existing graph transformation engine like AGG [AGG06] or VIATRA2 [BV06].

In [BK06] the foundations of rule-based transformations in EMF are described and applied using the AGG graph transformation tool environment [AGG06]. Basically, an EMF transformation is a rule-based modification of an EMF source model resulting in an EMF target model. Both, the EMF source and target models are typed over EMF metamodels which itself are again typed over Ecore. The transformation rules are typed over a *transformation metamodel* (*Transformation Meta* in Fig. 2) which is independent from the used transformation tool, but somehow dependent on the transformation approach. The trans-

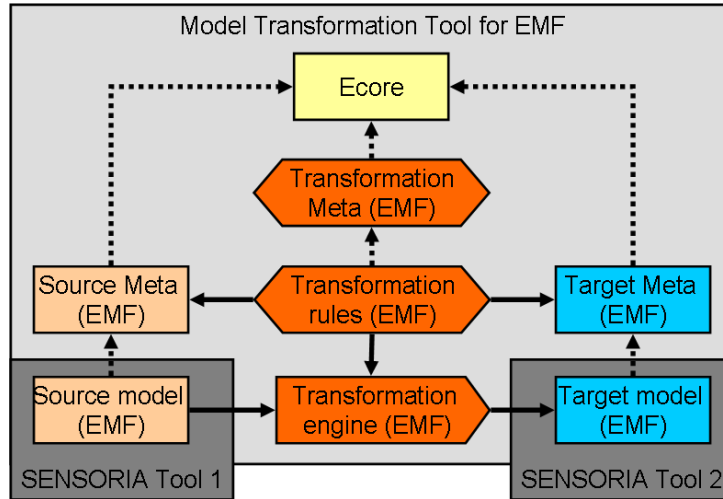


Figure 2: Interpreted model transformations for EMF

formation metamodel itself is an instance of Ecore again (see Fig. 2 where the *typed over* relations are indicated by dashed arrows). In the following, we describe the core concepts for transformation systems, inspired by graph transformation concepts.

In general a transformation system consists of a set of transformation rules. Rules are expressed mainly by two object structures LHS and RHS, the left and right-hand sides of the rule. Furthermore, a rule has a mapping between LHS and RHS patterns. The left-hand side LHS represents the pre-conditions of a rule, while the right-hand side RHS describes its post-conditions. Those pattern parts of the LHS which are mapped to the RHS, describe a structure part which has to occur in the EMF source model, but which is not changed during the transformation. All symbols and links of the LHS not mapped to the RHS define the part which shall be deleted, and all symbols and links of the RHS which are not used for mapping, define the part to be created.

The applicability of a rule can be further restricted by additional application conditions. As already mentioned above, the LHS of a rule formulates some kind of positive condition. In certain cases also *negative application conditions* (NACs) which are pre-conditions prohibiting certain object structures, are needed. If several NACs are formulated for one rule, each of them has to be fulfilled.

Performing a transformation step which applies a rule at a selected match, the resulting object structure is constructed in two passes: (1) all objects and links present in the LHS but not in the RHS are deleted; (2) all object and links in the RHS but not in the LHS are created. A transformation, more precisely a transformation sequence, consists of zero or more transformation steps.

To apply the defined transformation rules to a given EMF model, we either select and apply the rules step-by-step, or take the whole rule set and let it apply as long as possible. A transformation step with a selected rule is defined by first finding a match of the left-hand side in the current instance model. A pattern matches to a model if its structure can be found in the model such that the types and attribute values are compatible. In general, a pattern can match different parts of a model. In this case, one of the



possible matches is selected, either randomly or by the user. Certain graph transformation approaches (e.g. VIATRA2 [BV06]) also allow to apply a rule on all matches in parallel in such a case.

Consistency recovery: Although EMF models show a graph-like structure and can be transformed similarly to graphs [EEPT06], there is a main difference between both concepts. In contrast to graphs, EMF models have a distinguished tree structure which is defined by the containment relation between their classes. An EMF model should be defined such that all its classes are transitively contained in the root class. Since an EMF model may have non-containment references in addition, the following question arises: What if a class which is transitively contained in the root class, has non-containment references to other classes not transitively contained in the root class? In this case, we consider the EMF model to be inconsistent, thus e.g. it cannot be persisted anymore.

A transformation can make an EMF model inconsistent, if its rule deletes one or more objects. For example an inconsistent situation occurs, if one of these objects transitively contains an object included by a non-containment reference. To restore the consistency, all objects to be deleted have to be determined. Thereafter, all non-containment references to these indicated objects have to be removed, too.

Similarly to the handling of deleted structures, consistency recovery is also applied to newly created objects. If a rule creates objects which are not contained in the tree structure, the consistency recovery will remove these objects at the end of a rule application. It is possible to forbid the application of those rules entirely, since inconsistencies on creation of objects can be determined statically.

5 Compiled Transformer Plug-ins

The third approach (illustrated in Fig. 3) for integrating model transformations over EMF models is the compilation of model transformation specification into a stand-alone transformer plug-in (e.g. a Java program), which takes a source EMF instance as input and generates a target EMF instance as output.

In the EMF context, such transformer plug-ins generated from graph transformation rules are basically Java programs, which manipulate EMF model instances via the domain-specific interfaces (APIs) generated by EMF. From a conceptual point of view, this compilation process needs to address the following main issues:

1. how to perform graph pattern matching efficiently in the case of single, parallel or as long as possible rule applications;
2. how to check the success or failure of pattern matching
3. how to manipulate models in a consistent and transactional way

It is well-known that the most critical step for the performance of graph transformation is the graph pattern matching phase. For this purpose, *constraint-solving techniques* and *search plan generation* are the two most frequently used and efficient strategies.

- 30
- Pattern matching can be formulated as a *constraint solving problem* where the LHS objects are variables, the objects of the EMF instance model form the domain and typing, linking und attribute values form the set of constraints.

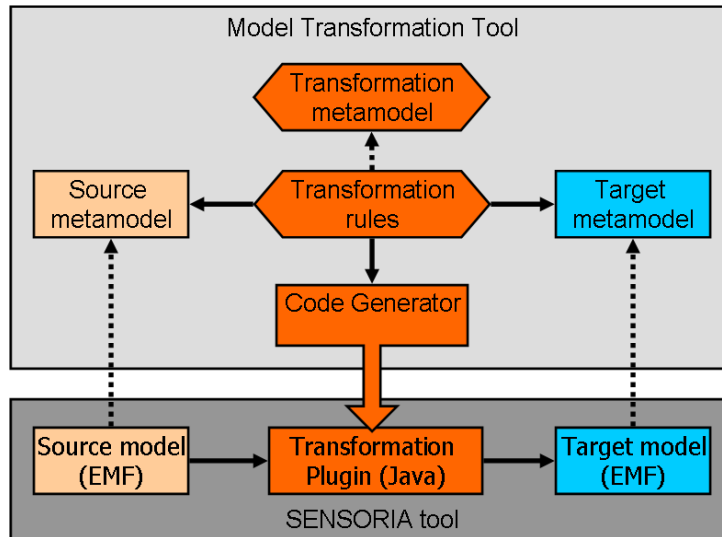


Figure 3: Transformer plugins in Java

- A *search plan* defines (at compilation time) the traversal order for the nodes of the instance model to check whether the pattern can be matched. Complex model-specific optimization steps can be carried out for generating efficient adaptive search plan as reported in [VVF05].

For rule application, Java code is generated which relies on those EMF classes already generated from the EMF model.

Transformation services by EJB3 transformer plugins. While EMF models are becoming increasingly popular, large service-oriented systems typically use other Java-based technologies providing persistent storage for business data and transaction handling for business transactions. Fortunately, the interfaces provided by EMF are very close to those Java-based techniques being e.g. Enterprise Java Beans (EJB), which are frequently used in service-oriented and web-applications. Therefore, we also investigated in [BVVP06, Var06] how leading industrial Java technologies (like EJB 3.0 [Sun06]) can be used as a platform for transformer plugins.

- **Compiling EJB3 models.** *EJB3 entity bean classes* will be generated from the source and target metamodels including the reference objects representing the mapping between them. The persistent storage of EJB3 entity beans will then be handled by the EJB3 application server.
- **Compiling EJB3 transformations.** *EJB3 session beans* can be generated from graph transformation rules in the form of fully functional EJB business methods. Transaction handling will be provided by the application server to prevent complex transformations from introducing conflicts when manipulating the model in parallel.

- **Compiling EJB3 search plans.** The concepts of search plans were adapted to the underlying database technology by translating graph pattern matching problems into database-independent EJB-QL queries, which reduce the memory consumption of traversing large model graphs.

Our measurements in [BVVP06] have demonstrated that transformer plugins with an underlying persistent store outperform pure Java solutions considering the size of the graph model by enabling to transform several million graph objects.

Since model transformations carried out by EJB3 transformer plugins can be easily published as web services, they fit very well to the service-oriented SENSORIA approach.

6 Related Work

There are several model transformation tools based on the Eclipse technology on the one hand, and on graph transformation techniques on the other hand. In [TEG⁺05] we present a comparative study about different model transformation by graph transformation approaches. Furthermore, we started a first comparison with QVT [QVT05], the proposal for model transformation given by the OMG. In the following we distinguish between QVT-related and graph transformation related tools.

QVT-Related Tefkat [LS05] implements a declarative model transformation language for the transformation of MOF models using patterns and rules. It is implemented as an Eclipse plugin using the Eclipse Modeling Framework (EMF) to handle models based on MOF, UML2, and XML Schema. Unlike XSLT, Tefkat has a simple and familiar looking SQL-like syntax. It is specifically designed for writing scalable and re-usable transformation specifications using high-level domain concepts rather than operating directly on XML syntax.

ATL [ATL06] is a model transformation language specified both as a metamodel and as a textual concrete syntax. It is a hybrid of declarative and imperative approaches. The preferred style of transformation writing is declarative, which means simple mappings can be expressed simply. However, imperative constructs are provided so that some mappings which are getting too complex to be declaratively handled, can still be specified. Once complex mapping patterns are identified, declarative constructs can be added to ATL in order to simplify the writing of transformation systems. An IDE has been developed for ATL on top of Eclipse: ATL Development Tools (ADT), which uses EMF to handle models.

The Merlin [Mer06] Eclipse plug-in is based on the EMF JET Templates & Mapping model whose goal is to ease the process of automating the code generation and model transformation.

The Model Transformation Framework (MTF) [MTF05] is a set of tools that helps developers to make comparisons, check consistency, and implement transformations between Eclipse Modeling Framework (EMF) models. The framework also supports persistence of a record of what was mapped to what by the transformation; this record can be used to support round-tripping, reconciliation of changes, or display of the results to a user.

The MOMENT project contains a QVT-like EMF transformation engine which is based on algebraic specifications as implemented in Maude [BCR06]. This approach has a clear formal background which can be used for verification, but does not yet provide a visual definition of transformations.



In contrast to most of the related approaches, EMF transformations can be given a formal background by graph transformation which can be advantageously used to analyze model transformations. Analysis techniques include conflicts and dependencies analysis of rule applications, termination of model transformations and constraint checking.

Graph Transformation Related The Visual Modeling and Transformation System (VMTS) [LLMC04] is an n-layered metamodeling environment designed together with model transformation functionalities. VMTS is offering capabilities for specifying visual languages applying metamodeling techniques.

The AToM³ [dLV02] tool allows the specification of Domain Specific Visual Languages by means of meta-modelling, and their manipulation by means of graph transformation. Recently, AToM³ has been provided with the possibility to define triple graph grammars [GdL04] and multiple views [GDdL05]. Views are provided with their own meta-model, which optionally can be a subset of a global meta-model that relates all the view concepts. This is very useful when defining multi-view languages (specially if such views contain overlappings), such as UML.

GReAT (Graph Rewriting And Transformation) [GRe06] is a metamodel based graph transformation language useful for the specification and implementation of model-to-model transformations.

MoTMoT [Mot06] stands for Model driven, Template based, Model Transformer. It is a compiler from visual model transformations to repository manipulation code. The compiler takes models conforming to a UML profile for Story Driven Modeling (SDM) [FNTZ98] as input and outputs JMI code.

In [SG04] SDM has been used as a language for the visual development of refactorings (which are a particular kind of horizontal model transformations) and implemented in Fujaba [Fuj06] which suffers from two significant problems. First, the SDM metamodel in Fujaba is non-standard and it is only implicitly present in the source code. As a consequence, only the Fujaba editor is suitable to create and store SDM instances. Second, the Fujaba code generator exclusively generates code conforming to non-standard conventions, meaning it can solely be deployed on the Fujaba repository.

7 Conclusion

In this paper, we proposed three approaches for tool integration based on EMF model transformations. The following levels of tool integration were identified: (i) *model-level integration* carries out model transformations in existing transformation tools by importing and exporting EMF models, (ii) *interpreted EMF transformations* take a model of the transformation system, and manipulate EMF models according to the its rules by calling EMF interfaces, and finally (iii) *compiled transformer plugins* generate stand-alone transformer programs in Java which are responsible for model manipulation.

In SENSORIA, two main transformation tools, namely, TIGER [EEHT05] and VIATRA2 are planned to be used for tool integration purposes using our approach. While VIATRA2 is mainly a model transformation tool, the main purpose of TIGER is the generation of visual language environments from high-level specifications. Currently, the generation of visual editors and the execution of model transformations are supported. While VIATRA2 has support for approaches (i) and (iii), TIGER will have support for 3 approaches (ii) and (iii), based on the EMF transformer developed in [BK06].



Acknowledgements

This work was partially supported through the IST-2005-16004 Integrated Project SENSORIA: Software Engineering for Service-Oriented Overlay Computers. The third author also received partial support from the János Bolyai Scholarship.

References

- [AGG06] *AGG-System* <http://tfs.cs.tu-berlin.de/agg/>, 2006.
- [ATL06] *ATL: The Atlas Transformation Language Home Page* <http://www.sciences.univ-nantes.fr/lina/atl>, 2006.
- [BCR06] A. Boronat, J. Carsi, and I. Ramos. Algebraic Specification of a Model Transformation Engine. In *Springer LNCS 3922. Fundamental Approaches to Software Engineering (FASE'06). ETAPS'06. Vienna (Austria)*, 2006.
- [BK06] Enrico Biermann and Günter Kuhns. Konzeption und Implementierung einer regelbasierten Transformationskomponente für das Eclipse Modeling Framework. Master's thesis, Technical University of Berlin, Department of Computer Science, 2006.
- [BV06] András Balogh and Dániel Varró. Advanced model transformation language constructs in the VIATRA2 framework. In *ACM Symposium on Applied Computing — Model Transformation Track (SAC 2006)*, pages 1280–1287, Dijon, France, April 2006. ACM Press.
- [BVVP06] András Balogh, Gergely Varró, Dániel Varró, and András Pataricza. Compiling model transformations to EJB3-specific transformer plugins. In *ACM Symposium on Applied Computing — Model Transformation Track (SAC 2006)*, pages 1288–1295, Dijon, France, April 2006. ACM Press.
- [dLV02] J. de Lara and H. Vangheluwe. AToM3: A tool for multi-formalism and meta-modelling. In R.-D. Kutsche and H. Weber, editors, *5th International Conference, FASE 2002: Fundamental Approaches to Software Engineering, Grenoble, France, April 8-12, 2002, Proceedings*, volume 2306 of *Lecture Notes in Computer Science*, pages 174–188. Springer-Verlag, 2002.
- [EEHT05] K. Ehrig, C. Ermel, S. Hänsgen, and G. Taentzer. Generation of visual editors as eclipse plugins. In *Proc. 20th IEEE/ACM International Conference on Automated Software Engineering*, IEEE Computer Society, Long Beach, California, USA, 2005.
- [EEPT06] H. Ehrig, K. Ehrig, U. Prange, and G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. EATCS Monographs in Theoretical Computer Science. Springer, 2006.
- [EMF06] *Eclipse Modeling Framework (EMF)* <http://www.eclipse.org/emf>, 2006.
- ³⁴[EMO06] *Essential MOF (EMOF) as part of the OMG MOF 2.0 specification* <http://www.omg.org/docs/formal/06-01-01.pdf>, 2006.



- [FNTZ98] T. Fischer, J. Niere, L. Torunski, and A. Zündorf. Story Diagrams: A New Graph Rewrite Language Based on the Unified Modeling Language and Java. In *In Proceedings of the 6th International Workshop on Theory and Application of Graph Transformation (TAGT), volume 1764 of LNCS, pages 296-309*. Springer, 1998.
- [Fuj06] *Fujaba Project*, 2006. Available at <http://www.fujaba.de>.
- [GDdL05] E. Guerra, P. Díaz, and J. de Lara. A Formal Approach to the Generation of Visual Language Environments Supporting Multiple Views. In *Proc. IEEE Symposium on Visual Languages and Human-Centric Computing, IEEE VL/HCC, Dallas, 2005*.
- [GdL04] E. Guerra and J. de Lara. Event-Driven Grammars: Towards the Integration of Meta-Modelling and Graph Transformation. In *In Proc. ICGT'04 (Rome). LNCS 3256, pp.: 54-69*. Springer, 2004.
- [GRe06] *GReAT: Graph Rewriting And Transformation* <http://www.isis.vanderbilt.edu/Projects/mobies/downloads.asp>, 2006.
- [LLMC04] T. Levendovszky, L. Lengyel, G. Mezei, and H. Charaf. Systematic Approach to Meta-modeling Environments and Model Transformation Systems in VMTS. In *2nd International Workshop on Graph Based Tools (GraBaTs), workshop at ICGT 2004, Rome, Italy, 2004*.
- [LS05] M. Lawley and J. Steel. Practical Declarative Model Transformation With Tefkat. In *In Proc. Model Transformation in Practice Workshop, Models Conference, 2005*.
- [Mer06] *Merlin Generator* <http://sourceforge.net/projects/merlingenerator/>, 2006.
- [Mot06] *MoTMoT: Model driven, Template based, Model Transformer* <http://www.fots.ua.ac.be/motmot/index.php>, 2006.
- [MTF05] *IBM Model Transformation Framework* <http://www.alphaworks.ibm.com/tech/mtf>, 2005.
- [QVT05] *Query/View/Transformation (QVT). QVT-Merge Group, version 2.0 (2005-03-02)*. <http://www.omg.org/cgi-bin/apps/doc?ad/05-03-02.pdf>, 2005.
- [SG04] Hans Schippers and Pieter Van Gorp. Standardizing SDM for Model Transformations. In *Second Int. Fujaba Days (FD04), Darmstadt (Germany), 2004*. Available at <http://www.lore.ua.ac.be/refactoringProject/publications/StandardizingS%DMforModelTransformations.pdf>.
- [Sun06] Sun Microsystems. *Enterprise Java Beans 3.0*, 2006. <http://java.sun.com/products/ejb/docs.html>.



- [TEG⁺05] G. Taentzer, K. Ehrig, E. Guerra, J. de Lara, L. Lengyel, T. Levendovsky, U. Prange, D. Varro, and S. Varro-Gyapay. Model Transformation by Graph Transformation: A Comparative Study. In *Proc. Workshop Model Transformation in Practice*, Montego Bay, Jamaica, October 2005.
- [Var06] Gergely Varró. Implementing an EJB3-specific graph transformation plugin by database independent queries. In *Proc. of the Fifth International Workshop on Graph Transformation and Visual Modelling Techniques*, ENTCS, pages 115–126. Elsevier, 2006.
- [VIA06] VIATRA2 (Visual Automated model TRAnsformations) framework [http://dev.eclipse.org/viewcvs/indextech.cgi/~checkout~/gmt-home/subpro%
jects/VIATRA2/index.html](http://dev.eclipse.org/viewcvs/indextech.cgi/~checkout~/gmt-home/subprojects/VIATRA2/index.html), 2006.
- [VP04] Dániel Varró and András Pataricza. Generic and meta-transformations for model transformation engineering. In T. Baar, A. Strohmeier, A. Moreira, and S. Mellor, editors, *Proc. UML 2004: 7th International Conference on the Unified Modeling Language*, volume 3273 of LNCS, pages 290–304, Lisbon, Portugal, October 10–15 2004. Springer.
- [VVF05] Gergely Varró, Dániel Varró, and Katalin Friedl. Adaptive graph pattern matching for model transformations using model-sensitive search plans. In G. Karsai and G. Taentzer, editors, *GraMoT'05, International Workshop on Graph and Model Transformations*, ENTCS Vol. 152, 2005.

Minutes of the EASST General Assembly

March 27, 2006, Vienna

The EASST General Assembly in 2006 took place on Monday 27th March at 18.00 at the TU Vienna (Austria) in the FASE conference room and it was chaired by the EASST President Tiziana Margaria.

A list of the association members is passed around to be signed by the present members, which is enclosed as a separate document.

The EASST president opens the works by asking the assembly to approve the agenda, which is done with no additions.

TOP1: Reports by the Board Members

Then, she starts the report from the board members with the announcement that the list of actions of the Board Meeting 2005 has been addressed in all points, with particular mention to the settlement of financial affairs. An account at the Postbank Berlin has been opened and is operational, and a reformulation of ambiguous or imprecise points of the Statute has been carried out in order to be presented to this General Assembly for ratification (TOP 3). The aim is to get the status of association of public interest during 2006.

Reports by all the Board members have been provided in writing and are therefore not orally presented.

TOP 2 Reports by the National Representatives and Topic Representatives

Reports by the National and Topic Representatives have been provided in writing and are therefore not orally presented. They will be included in the Summer issue of the EASST Newsletters.

TOP 3 Ratification of the new EASST statute

The General Assembly approves unanimously the new EASST statute.

TOP 4 Election of the new organs (Chaired by Susanne Graf)

The General Assembly has elected unanimously the following Organs:

EASST President: Tiziana Margaria
EASST Vice President: Hartmut Ehrig
EASST Treasurer: Michael G. Hinchey

The elected accepted the election during the meeting (T. Margaria and H. Ehrig) and per e-mail (M. Hinchey).

The Assembly thanks Prof. Dr. Herbert Weber, the outgoing Treasurer and first EASST President, for his activities and engagement for EASST since its inception.

TOP 5 Plans for the coming year

The General Assembly mandates the EASST Board to continue in the directions taken, taking particular care of education-related issues.

Others

No other matters were discussed

Next meeting:

The next EASST General Assembly is planned for ETAPS 2007 in Braga, Portugal

End of Minutes

Vienna, April 1st, 2006

Tiziana Margaria
(Protocol)

Reports of Members of the Extended Board

Jens Knoop

- **Service to EASST, since 04/2004:**
 - EASST Topic Representative for Software Analysis and Verification,
 - EASST Extended Boards Member and
 - EASST Newsletter, Editor of The Software Analysis and Verification Column.
- **Service to the EASST Newsletter**
 - The Software Analysis and Verification Column.
The EASST Newsletter, Volume 10, No. 1, June 2005, 23 – 24.
 - The Software Analysis and Verification Column.
The EASST Newsletter, Volume 9, No. 2, December 2004, 9 - 12.
- **Service to EASST Sponsored or Endorsed Conferences**
 - 9th European Joint Conferences on Theory and Practice of Software (ETAPS 2006) (Vienna, Austria, March 25 - April 2, 2006), General Chair.
 - 1st International EASST-EU Workshop on Future Research Challenges for Software and Services (FRCSS 2006), (Vienna, Austria, April 1, 2006). In conjunction with the 9th Joint European Conferences on Theory and Practice of Software (ETAPS 2006), (Vienna, Austria, March 25 - April 2, 2006), Programme Committee Member.
 - 1st International Symposium on Leveraging Applications of Formal Methods (ISoLA 2004) (Paphos, Cyprus, October 30 - November 2, 2004), Thematic Topic Chair, Programme Committee Member.

Service to Further EASST Related Conferences and Seminars

a) Organization

- 5th International Workshop on Compiler Optimization Meets Compiler Verification (COCV 2006), (Vienna, Austria, April 2, 2006). In conjunction with the 9th Joint European Conferences on Theory and Practice of Software (ETAPS 2006), (Vienna, Austria, March 25 - April 2, 2006), Co-Chair.
- Dagstuhl-Seminar 05311 on Verifying Optimizing Compilers. July 31 - August 5, 2005, Co-Organizer.
- 4th International Workshop on Compiler Optimization Meets Compiler Verification (COCV 2005), (Edinburgh, Scotland, April 3, 2005). In conjunction with the 8th Joint European Conferences on Theory and Practice of Software (ETAPS 2005), (Edinburgh, Scotland, April 2 - 10, 2005), Co-Chair.
- 3rd International Workshop on Compiler Optimization Meets Compiler Verification (COCV 2004), (Barcelona, Spain, April 3, 2004). In conjunction with the 7th Joint European Conferences on Theory and Practice of Software (ETAPS 2004), (Barcelona, Spain, March 27 - April 4, 2004), Co-Chair.

b) Publications

- Report on the Dagstuhl-Seminar 05311 on Verifying Optimizing Compilers. Schloss Dagstuhl, Wadern, Germany, Co-Editor.
- Proceedings of the 4th International Workshop on Compiler Optimization Meets Compiler Verification, in conjunction with ETAPS 2005, Electronic Notes in Theoretical Computer Science ENTCS, Volume 141, Issue 2, Pages 1-120 (7 December 2005), Co-Editor.
- Proceedings of the 3rd International Workshop on Compiler Optimization Meets Compiler Verification, in conjunction with ETAPS 2004, Electronic Notes in Theoretical Computer Science ENTCS, Volume 132, Issue 1, Pages 1-148 (30 May 2005), Co-Editor.

- Preliminary Proceedings of the International Symposium on Leveraging Applications of Formal Methods (ISoLA 2004), Thematic Session on Program Analysis and Transformation, Department of Computer Science, University of Cyprus, TR-2004-6, 1 - 2, 2004.
- Service to Forthcoming EASST Sponsored/Related Events
 - 6th International Workshop on Compiler Optimization Meets Compiler Verification (COCV 2007), (Braga, Portugal, March 25, 2007, tentatively). In conjunction with the 10th Joint European Conferences on Theory and Practice of Software (ETAPS 2007), (Braga, Portugal, March 24 - April 1, 2007), Co-Chair.
 - 2nd International Symposium on Leveraging Applications of Formal Methods, Verification, and Validation (ISoLA 2006) (Paphos, Cyprus, November 15 - 19, 2006, tentatively), Track Chair.
 - 4th International Conference in Central Europe on .NET Technologies (.NET Technologies 2006) (Plzen, Czech Republic, May 29 - June 1, 2006), Co-Chair

Marie-Claude Gaudel

It turns out that we significantly improved the interface with PC chairs this year:

The information letter that we prepared for them, in Edinburgh last year, is now posted on the web site. It was sent to them in December, just after the end of the selection process, when they were with fresh memory of it.

We had a number of nominations (6) for the ETAPS'06 award:

- One for CC: Context-sensitive points-to analysis: is it worth it?, by Ondrej Lhotak and Laurie Hendren.
- One for ESOP: Path Optimization in Programs and its Application to Debugging, by Akash Lal, Junghee Lim, Marina Polishchuk, Ben Liblit.
- Two for FASE: Relation of Code Clones and Change Couplings by Beat Fluri, Reto Geiger, Harald C. Gall, Martin Pinzger; and A Programming Language for Service Implementation by Dominic Cooney, Marlon Dumas, Paul Roe.
- Two for TACAS: Evaluating the Effectiveness of Slicing for Model Reduction of Concurrent Object-Oriented Programs, by Matthew B. Dwyer, John Hatcliff, Matthew Hoosier, Venkatesh Ranganath, Robby, Todd Wallentine ; and Exploiting Traces in Program Analysis, by Alex Groce and Rajeev Joshi.

As usual, the committee was the EASST board.

Just before the Christmas break, the members received: the nominated papers, the review reports and, for each paper, a short statement by the PC chair explaining the reasons for nomination. The members were asked to give their two best choices.

Then there was a debate by email and the final decision was in favor of:

A Programming Language for Service Implementation by Dominic Cooney, Marlon Dumas, Paul Roe.

We sincerely congratulate the authors.

We also sincerely thanks XXX who accepted to continue to support the EASST award allowing us continue the tradition of accompanying the award with a check.

Reiko Heckel

As thematic representative in charge of Visual Modelling Techniques, Reiko Heckel is regularly editing the column of the same name, providing a mix of scientific contributions and news from inside the SegraVis network, a European Research Training Network on Syntactic and Semantic Integration of Visual Modelling Techniques.

Among the events in the near future organised or supported by this network there are:

- The Advanced School on Visual Modelling Techniques, 8 - 11 September 2006 at the University of Leicester, UK (still accepting applications, see <http://www.cs.le.ac.uk/events/segravis/>);
- The International Conference on Graph Transformation, 17-22 September 2006, Natal, Brazil (CFPs of several satellite workshops are still open, see <http://www.dimap.ufrn.br/icgt2006/>)

Another important event in the area, also endorsed by EASST, is the 9th International Conference on Model Driven Engineering Languages and Systems, October 1-6 in Genova, Italy.

Reiko Heckel was co-chair of FASE 2006, one of the ETAPS conferences supported by EASST. A report on the conference can be found in this newsletter.

Julia Padberg

As the editor of the newsletter I can report on the two regular issues in June and December 2005. Moreover, there has been prepared a special issue of the newsletter for the proceedings of FRCSS 2006. This has been the starting point to revive the EASST activities for an open access journal (see the Welcome of this newsletter).



Report on ETAPS 2006 — The 9th European Joint Conferences on Theory and Practice of Software

Jens Knoop, ETAPS 2006 General Chair *

*Institute of Computer Languages, Vienna University of Technology

From 25th March to 2nd April 2006 Austria's capital city Vienna was the venue of *ETAPS 2006*, the *9th European Joint Conferences on Theory and Practice of Software*. ETAPS 2006 was hosted by the Vienna University of Technology and took place at the Elektrotechnisches Institut which is conveniently located in the heart of Vienna, just a 10 minutes walk away from the famous Viennese State Opera, the St. Stephen's Cathedral, and the Hofburg, the former Imperial Palace of the Habsburgs in downtown Vienna, originally a medieval castle, which later was extended to the magnificent residence we can see today, and which was the seat of the Austrian dynasty for more than 6 centuries.

This year's ETAPS conference was the 9th event in the successful ETAPS conference series, whose inaugural meeting was in Lisbon in 1998, which then was followed by annual meetings in Amsterdam, Berlin, Genova, Grenoble, Warsaw, Barcelona, and most recently in Edinburgh.

ETAPS 2006 was organized by the Compilers and Languages Group of the Institute of Computer Languages of the Vienna University of Technology together with the Austrian Computer Society (OCG) and the sponsoring organizations of the ETAPS conference series, the European Association of Programming Languages and Systems (EAPLS), the European Association for Software Science and Technology (EASST), and the European Association of Theoretical Computer Science (EATCS).

In addition to the 5 member conferences of ETAPS,

- CC: International Conference on Compiler Construction,
- ESOP: European Symposium on Programming,
- FASE: Formal Aspects of Software Engineering,
- FoSSaCS: International Conference on Formal Aspects of Software Science and Computation Structures, and
- TACAS: International Conference on Tools and Algorithms for the Construction and Analysis of Systems,

ETAPS 2006 featured 18 satellite events and 2 tutorials, which took place at the two weekends right before and after the ETAPS member conferences.

- ACCAT: Applied and Computational Category Theory,

- AVIS: Automated Verification of Infinite-State Systems,
- CMCS: Coalgebraic Methods in Computer Science,
- COCV: Compiler Optimization Meets Compiler Verification,
- DCC: Designing Correct Circuits,
- EAAI: Emerging Applications of Abstract Interpretation,
- FESCA: Formal Foundations of Embedded Software and Component-Based Software Architectures,
- FRCSS: Future Research Challenges for Software and Services,
- GT-VMT: Graph Transformation and Visual Modeling Techniques,
- LDTA: Language Descriptions, Tools and Applications,
- MBT: Model Based Testing,
- QAPL: Quantitative Aspects of Programming Languages,
- SC: Software Composition,
- SLAP: Synchronous Languages, Applications, and Programming,
- SPIN: Model Checking of Software,
- TERMGRAPH: Term Graph Rewriting,
- WITS: Issues in the Theory of Security, and
- WRLA: Rewriting Logic and its Applications.

The 2 tutorials, finally, were on *Quantum Information Processing* and the *Phoenix Framework for Code Generation and Program Analysis*. In more detail:

- *The Computer Science Perspective on Quantum Information Processing and Communication*, by Philippe Jorrand (Leibniz Laboratory, Grenoble, France), and
- *Phoenix: A Framework for Code Generation and Program Analysis*, by Chuck Mitchell (Microsoft), and Mark Lewin (External Research and Programs, Microsoft Research).

Altogether, the member conferences of ETAPS 2006 and the satellite events and tutorials co-located with it attracted more than 700 attendees from all over the world. The largest group of attendees this year came from the United States of America. Next to it were the groups of attendees from Germany, the UK, France, and Italy, however, there were also participants from countries as far away as Australia, Brazil,

South-Africa, Korea, and Japan. In total, there were around 500 participants attending the member conferences of ETAPS and some of the satellite events and tutorials co-located with it, and around 200 participants attending some of the satellite events and tutorials only.

The number of submissions to all member conferences of ETAPS 2006 was again very high, allowing the programme committees of the conferences to be very selective. In fact, the selection process was highly competitive with acceptance rates as low as 17%. In more detail, CC received this year a total of 71 submissions. Its programme committee accepted 17 research papers and 3 tool demonstration papers for presentation at the conference. This amounts to an acceptance rate of 28%. ESOP received 87 submissions out of which 21 were accepted for presentation at the conference. This corresponds to an acceptance rate of 24%. FASE received 166 research papers and 7 papers on tool demonstrations. The 27 research papers and 2 tool demonstration papers accepted make up an acceptance rate of 17%. FoSSaCS received 107 submissions. 28 papers were accepted for presentation at the conference. This corresponds to an acceptance rate of 26%. TACAS, finally, received 118 research papers and 9 tool demonstration papers, out of which 30 research papers and 4 tool demonstration papers were accepted for presentation at the conference. This amounts to an acceptance rate of 27%.

As in the previous years the proceedings of the member conferences of ETAPS were published in the *Lecture Notes of Computer Science (LNCS)* Series of Springer-Verlag.

- LNCS 3920, (Proc. 12th TACAS'06), (Eds. Holger Hermanns and Jens Palsberg)
- LNCS 3921, (Proc. 9th FoSSaCS'06), (Eds. Luca Aceto and Anna Ingólfssdóttir)
- LNCS 3922, (Proc. 9th FASE'06), (Eds. Luciano Baresi and Reiko Heckel)
- LNCS 3923, (Proc. 15th CC'06), (Eds. Alan Mycroft and Andreas Zeller)
- LNCS 3924, (Proc. 15th ESOP'06), (Ed. Peter Sestoft)

The proceedings of the SPIN workshop, edited by Antti Valmari, appeared in the LNCS Series, too, volume number LNCS 3925. The SC workshop and the WITS workshop are going to prepare post-conference proceedings. These proceedings will also be published in the LNCS series. The proceedings of most of the other satellite events at ETAPS 2006 will be published as a special issue in the *Electronic Notes of Theoretical Computer Science (ENTCS)* journal of Elsevier B. V.

Special highlights of the scientific programme were the presentations of the invited speakers at the ETAPS member conferences. These keynote speeches were delivered by *George Necula* (University of California, Berkeley, USA), the invited speaker at CC, *Sophia Drossopoulou* (Imperial College London, UK), the invited speaker at ESOP, *Francisco Curbera* (IBM TJ Watson, USA), the invited speaker at FASE, *Wan Fokkink* (Vrije Universiteit Amsterdam, NL), the invited speaker at FoSSaCS, and *Somesh Jha* (University of Wisconsin, Madison, USA), the invited speaker at TACAS. Moreover, *Carlo Ghezzi* (Politecnico di Milano) and *Benjamin Pierce* (University of Pennsylvania, USA) were the two invited unifying keynote speakers at this year's ETAPS conference. The topics of their keynotes were as follows:

- *Types for Hierarchic Shapes* by Sophia Drossopoulou on Monday (ESOP),
- *A Programming Model for Service Oriented Applications* by Francisco Curbera on Tuesday (FASE),

- *Software Engineering: Emerging Goals and Lasting Problems* by Carlo Ghezzi on Wednesday (First Unifying Invited Talk),
- *The Weird World of Bi-Directional Programming* by Benjamin Pierce on Wednesday (Second Unifying Invited Talk),
- *Distributed Model-Checking Algorithms for WPDS with Applications to Trust-Management Systems* by Somesh Jha on Thursday (TACAS),
- *Oh Mega Completeness* by Wan Fokkink on Thursday (FoSSaCS), and
- *Using Dependent Types to Port Type Systems to Low-Level Languages* by George Necula on Friday (CC).

All the invited speakers were interviewed after their presentation by the editor of the *Daily ETAPS*, Markus Schordan. These interviews were published in the very issue of the *Daily ETAPS* which appeared the day right after the presentation. All issues of the *Daily ETAPS* are ready for down-load on the homepage of ETAPS 2006 at www.complang.tuwien.ac.at/etaps06.

Another highlight at this year's ETAPS conference was the presentation of the *EAPLS*, *EASST*, and *EATCS Best Paper Awards*. This presentation took place at the main conference banquet in the Orangery at Schönbrunn Palace on Tuesday night.

- The EAPLS Best Paper Award was presented by Mark van der Brand to *Ben Rudiak-Gould* and *Alan Mycroft*, University of Cambridge, UK, and *Simon Peyton Jones*, Microsoft Research Cambridge, UK, for their paper on *Haskell is Not Not ML*, which was presented at ESOP on Monday.
- The EASST Best Paper Award was presented by Tiziana Margaria and Marie-Claude Gaudel to *Dominic Cooney*, *Marlon Dumas*, and *Paul Roe*, Queensland University of Technology, Australia for their paper on *GPSL: A Programming Language for Service Implementation*, which was presented at FASE on Monday.
- The EATCS Best Paper Award was presented by Fernando Orejas to *Lutz Schröder*, University of Bremen, Germany, for his paper on *A Finite Model Construction for Coalgebraic Modal Logic*, which was presented at FoSSaCS on Thursday.

Congratulations to all of them!

Last but not least, there shall be a note on the social programme at ETAPS 2006. As usual, the social programme was opened on Sunday of the first weekend with the Joint Pre-Conferences Workshops Dinner. The venue of this dinner was the *Piaristenkeller*, a 300-year-old monastery cellar, which featers besides the historic restaurant a wine cellar and the Emperor Franz Joseph Hat Museum. Before enjoying the dinner the attendees visited both the wine cellar and the hat museum. Photos of this excursion can be found on the homepage of ETAPS 2006 at www.complang.tuwien.ac.at/etaps06.

The social programme continued on Monday night, when the *Bürgermeister of the Bundeshauptstadt Wien* invited the participants of ETAPS 2006 to a reception in the Grand Hall of the Vienna City Hall. Besides the typical Viennese snacks and drinks, which were served, also a duo of musicians added to the

live entertainment and enjoyment of the participants making the reception a pleasant opportunity for the participants to socialize with their fellow colleagues in a relaxed atmosphere.

The highlight of the social programme was certainly the main conference banquet on Tuesday night. The location of this banquet was the *Orangery* at *Schönbrunn Palace*, which in former times was the summer residence of the Imperial family in Vienna. Those, who arrived early for the banquet could take the opportunity to have a look at the splendid gardens of Schönbrunn Palace, which are a very fine example of baroque garden architecture.

The social event on Wednesday night, which again was open to all ETAPS participants, was a visit at the *Heurigen* “*Schübel-Auer*” in Wien-Nußdorf. A “Heurigen” is the Viennese term for the wine of the most recent grape harvest but also a (generic) name of the places where the wine together with local food specialties is served. Two chartered streetcars brought all ETAPS attendees to this location in Wien-Nußdorf.

On Thursday night the ETAPS participants were invited to a reception in the *Prechtl-Saal* of the *Vienna University of Technology*. This reception was sponsored in part by Intel Corporation, and offered another opportunity to enjoy Viennese hospitality.

The finale of the social programme was the Joint Post-Conferences Workshops Dinner in the *Lanner-Saal* of the *Vienna City Hall* on Saturday night at the second weekend of ETAPS 2006.

In addition to these events, there were two banquet dinners devoted to a specific workshop, the CMCS dinner on Saturday night at the first weekend, and the SPIN dinner on Friday night.

A conference like ETAPS cannot be organized without the support of many individuals and institutions. This year, ETAPS was happy to receive invaluable support by the following sponsors and partners:

- The *Institute of Computer Languages*,
- the *Faculty of Informatics of TU Vienna*,
- *TU Vienna*,
- the *Austrian Computer Society (OCG)*,
- the *European Association of Programming Languages and Systems (EAPLS)*,
- the *European Association for Software Science and Technology (EASST)*,
- the *European Association of Theoretical Computer Science (EATCS)*,
- *Intel Corporation*,
- the *Vienna Convention Bureau*, and
- the *Bürgermeister der Bundeshauptstadt Wien*.

In my capacity as the ETAPS 2006 General Chair, I would like to express my sincere thanks to all these institutions and their representatives. A special thanks goes to the members of my group of the Compilers and Languages Groups, especially to Andreas Krall, the workshops chair, to Franz Puntigam, the tutorials and web chair, and to Markus Schordan, the editor of the *Daily ETAPS*, and to the numerous student



helpers. Invaluable was also the help and support of the staff of the Austrian Computer Society (OCG). My special thanks belong to Eugen Mühlvenzl, the general secretary of the OCG, to Sandra Leitner, and Johann Stockinger, and all the other members of the OCG team, who did a marvellous job not only before the conference, but also on site during the conference. A special thanks also belongs to Christiane Tronigger from NetHotels Vienna, who did a perfect job in handling the registration and hotel reservation affairs for the conference. Thank you very much also to Alfred Hofmann and Ingrid Beyer from Springer-Verlag, and to Michael Mislove, the Managing Editor of ENTCS, for the smooth co-operation. Last but not least, I would like to thank the members of the Steering Committee of ETAPS, especially Perdita Stevens, the current chair of this committee, for their invaluable support and advice. My special thanks also goes to Don Sannella, whose information on previous ETAPS conferences, especially on ETAPS 2005 in Edinburgh of course, was extremely useful and helpful. I would also like to thank the Programme Committee Chairs of the member conferences of ETAPS 2006 as well as of the satellite events, the members of the programme committees and the external reviewers, the authors who submitted or even presented a paper at one of the conferences or workshops, and also all those who “just” attended the conference. Without their hard and excellent work, an event like ETAPS would not be possible at all. In particular, I would like to thank Ewa Vesely, my former secretary, who took a week of vacation to help on site of the conference, for their splendid work in preparing and organizing ETAPS 2006.

Please remember that photos taken at the conference, especially at the social events, are available for your enjoyment on the homepage of ETAPS 2006 at www.complang.tuwien.ac.at/etaps06.

Next year, ETAPS 2007 is going to return to Portugal. It will take place in Braga and will be organized by João Alexandre Saraiva and his team. Up-to-date information on this forthcoming 10th jubilee conference of ETAPS is available on the homepage of ETAPS 2007 at www.di.uminho.pt/etaps07/. Please check out this information and consider to submit a paper to a member conference of ETAPS 2007 or to a satellite event co-located with it.

I look forward to seeing you at ETAPS 2007 in Braga!



Report on the 9th Conference on Fundamental Approaches to Software Engineering

FASE 2006

Luciano Baresi* and Reiko Heckel**

*Politecnico di Milano, Italy

**University of Leicester, UK

***Abstract.** FASE, the Conference on Fundamental Approaches to Software Engineering, is held regularly as part of ETAPS, the European Conference on Theory and Application of Software. In its 9th installment ETAPS was organised in the city of Vienna in Austria, between the 25th of March and the 2nd of April 2006. This report summarises the objectives of FASE 2006 and gives an overview of the contributions, invited presentations, and points of general interest.*

Keywords: FASE 2006, ETAPS 2006

Software Engineering aims to create a feedback cycle between academia and industry, proposing new solutions and identifying those that “work” in practical contexts. As one of the European Joint Conferences on Theory and Practice of Software (ETAPS), the conference on Fundamental Approaches to Software Engineering (FASE) is committed to this aim.

With the society increasingly relying on software, the ability to produce low cost, high quality software systems is crucial to technological and social progress. FASE provides software engineers with a forum for the discussion of theories, languages, methods, and tools deriving from the interaction of academic research and real-world experience.

FASE 2006 was held in Vienna (Austria) as part of ETAPS 2006, hosted and organised by the Institute for Computer Languages at the Vienna University of Technology. Heartfelt thanks are due to Perdita Stevens for excellent and efficient global coordination of ETAPS and to Jens Knoop and his staff for their wonderful job as local organisers.

True to its profile at the interface of fundamental research and its application, FASE has been seeking submissions targeting problems of practical relevance on solid mathematical or conceptual foundations. The response of the scientific community was overwhelming, with record submission numbers of 166 research papers and 7 tool papers. From these, 27 research papers and 2 tool papers were selected for publication, with an overall acceptance ratio of 17%. We are deeply indebted to the 24 members of the programme committee and the 123 additional reviewers for their invaluable time, spent reading and



discussing a large number of papers and producing more than 500 reviews.

Accepted papers have addressed topics like distributed and service-oriented computing, measurement and empirical software engineering, methods and tools of software development, validation and verification, model-based development, and software evolution. The international character of the conference was underlined by the fact that just about one third of the authors were European, with others from North America, Asia and Australia.

The scientific programme was complemented by the invited lecture of Francisco Curbera on *A Programming Model for Service Oriented Applications*. As unifying invited lecture for the whole of ETAPS, FASE contributed the presentation of Carlo Ghezzi on *Software engineering: emerging goals and lasting problems*.

Francisco Curbera introduced the Service Component Architecture (SCA), a set of specifications which describe a model for building applications and systems using a Service-Oriented Architecture. SCA extends and complements prior approaches to implementing services, building on open standards such as Web services while focussing on the implementation of components which provide services and consume other services and the assembly of sets of components into business applications.

Carlo Ghezzi stressed the evolution of software from pre-defined, monolithic, centralised architectures to increasingly decentralised, distributed, dynamically composed federations of components and services. With development processes and organisations evolving along similar lines, this is affecting the way software is engineered, raising new difficult challenges, while old fundamental problems remain. The talk surveyed this evolution and identified achievements, challenges, and research directions.

With the best paper award of EASST being awarded to *GPSL: A Programming Language for Service Implementation* by Dominic Cooney, Marlon Dumas, and Paul Roe (see also the extended abstract in this newsletter), Service-oriented Computing was clearly the leading topics of the conference.

Generally, a trend towards an increasing number of contributions from mainstream software engineering could be perceived, addressing topics like software measurement and empirical software engineering, which have not received much attention at previous installments of FASE.

A noteworthy point of general interest has been the reception for ETAPS participants sponsored by Intel, indicating increasing interest of hardware providers in software science. This has to do with the expected slow-down of the rate of integration in microchip technology, which forces manufacturers to move on to different hardware architectures, requiring new programming models, languages, and compilers.

Other memorable events included a reception in the gorgeous neogothic city hall, the conference banquet in the orangery of Schönbrunn palace, and dinner at Heuriger, a traditional pub (where daring customers have been seen to mix wine with water ...).

Next year's FASE will take place in Braga, in the North of Portugal, as part of the 10th installment of ETAPS.

See you there!
Luciano and Reiko



Report on ACCAT Workshop at ETAPS 2006: Applied and Computational Category Theory

Hartmut Ehrig

Technical University of Berlin
Institute for Software Engineering and Theoretical Computer Science
Franklinstr. 28/29, D-10587 Berlin
ehrig@cs.tu-berlin.de

Category Theory is a well-known powerful mathematical modeling language with a wide area of applications in mathematics and computer science, including especially the semantical foundations of topics in software science and development. Since about 30 years there have been workshops including these topics. More recently, the ACCAT group established by J. Pfalzgraf at Linz and Salzburg has begun to study interesting applications of category theory in Geometry, Neurobiology, Cognitive Sciences, and Artificial Intelligence. It is the intention of this ACCAT workshop to bring together leading researchers in these areas with those in Software Science and Development in order to transfer categorical concepts and theories in both directions.

The ACCAT 2006 workshop was organized by Jochen Pfalzgraf and Hartmut Ehrig and took place on March 26, 2006 as satellite of ETAPS 2006 at the Vienna University of Technology. The organizers are representatives of categorical methods for several areas like Geometry, Neurobiology, Cognitive Sciences, and Artificial Intelligence on one hand and Software Science and Development on the other hand. Categorical methods are already well-established for the semantical foundation of type theory (Cartesian closed categories), data type specification frameworks (institutions) and graph transformation (adhesive high level replacement categories), which are most relevant for ETAPS. The organizers have invited leading senior and promising junior researchers for giving invited lectures at the ACCAT workshop which promises to lead to interesting discussions concerning transfer of categorical methods between the areas mentioned above.

After a short opening statement by the organizers Jochen Pfalzgraf continued with an interesting overview of the ACCAT origins. Julia Padberg reported about the integration of two important categorical frameworks, the generic component concept for system modelling and adhesive HLR systems, which is the subject of her habilitation thesis completed recently. After the presentation of a functional framework for constraint normal logic programming by Fernando Orejas we learned about a category theory in Brazil from Ciara Aparecida dos Santos Leal, especially a categorical view of structural complexity.



Andrzej Tarlecki and Till Mossakowski gave a very nice overview about institutions, abstract model theory, heterogeneous specification and the heterogeneous tool set. Motivated by homotopy theory in topology J. Rosicky reported about factorization systems and classification problems. Jose Meseguer discussed in his presentation different aspects of theory morphisms in membership equational logic.

The new topic of adhesive categories and adhesive high-level replacement systems was used by Ulrike Prange to present general constructions and properties of such categories and Andres Corradini discussed how to use this framework for a categorical semantics of concurrency generalizing that of graph transformation systems in the double pushout approach. Since Jiri Adamek and D. Dubois could not attend ACCAT we had enough time for long discussions of the presentations.

ACCAT will be continued next year as satellite workshop of ETAPS 2007 in Braga, Portugal.



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

Ralf H. Reussner ¹ * and Iman H. Poernomo **
and Juliana Küster-Filipe Bowles ***

*University of Karlsruhe (TH), Germany

**King's College, London, UK

***University of Birmingham, UK

***Abstract.** The FESCA 2006 workshop took place on Sunday 26th March, 2006 as a satellite event of ETAPS in Vienna. 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. The invited talk by Frantisek Plasil, Charles University Prague contributed to a highly successful event. Besides the discussion of the presented approaches, various kinds of component models and how to transfer insights from component models for enterprise systems to component models for embedded systems were discussed.*

Keywords: formal methods, component models, embedded systems

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. Formal methods have sometimes not kept up with the increasing complexity of software. For instance, a range of new middleware platforms have

¹Corresponding author: Institute for Program Structures and Data Organisation, University of Karlsruhe (TH), Am Fasanengarten 5, D-76131 Karlsruhe, Germany, reussner@ipd.uka.de



been developed in both enterprise and embedded systems industries. Often, engineers use semi-formal notations such as UML2 to model and organize components into architectures. FESCA aims to address the open question of how formal methods can be applied effectively to these new contexts and challenges. FESCA therefore is interested in formal methods known from the area of embedded software development and software engineering and tries to cross-fertilize their research and application.

One strength of FESCA is the link established between the embedded software design community and the formal software engineering community by exploring how formal approaches developed within one community affect or can be exploited by the other.

Previous FESCA workshops achieved this by looking at new computing paradigms like ubiquity, component-orientation and novel middleware technologies, which are of shared interest for both, embedded software design and formal software engineering.

Therefore, FESCA 2006 specifically called on other areas of shared interest:

Dependability: Due to safety-requirements common for many embedded systems, dependability research is often concerned with embedded software controlling technical systems. However, the demonstration of the dependability of a system is of increasing relevance also for enterprise software, as increasingly mission-critical enterprise systems and e-Commerce rely on software support.

Quality attributes and resource consumption: In both domains, embedded and enterprise software, quality attributes gain an increasing interest. While current software development processes, as used in industry, are mainly driven by the correct implementation of functional requirements, the systematic evaluation and prediction of quality attributes such as reliability, availability, resource consumption, performance and scalability is a matter of research. We consider formalisms used in one of the domains of embedded or enterprise software to be useful for the other. Given the complexity of today's concurrent, distributed and networked software, it is extremely important to provide formal techniques and CASE tools for analysis and reasoning on local component properties as well as on global system properties.

We encouraged specifically submissions on formal techniques that aid reasoning, analysis and certification of component-based embedded software and enterprise software.

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 2007 in Braga, Portugal.

The success of FESCA shows in the high quality of the submissions attracted. Common topics of the eight papers presented were component models and specification (including component contracts), component selection and model checking. Besides these presentations, FESCA gained attraction through our invited speaker Frantisek Plasil who presented joint work with Petr Hnetynka on hierarchical and flat component models.



2 Abstracts of the Contributions

In the following summaries of the presented papers are given.

Session I – Component Models and Specification

***ConCom - A Formal Model for Concurrent Components* by A. Rausch (Technical University of Kaiserslautern, Germany)**

Most software systems are structured in hierarchical components to manage complexity and have to cope with concurrent executed threads. Hierarchical system decomposition and concurrent flow of execution are orthogonal. A sound semantic model that is powerful enough to handle concurrent components but also realistic enough to provide a foundation for component technologies actually in use is still missing. Therefore, the paper introduces such an operational semantics for concurrent component-based systems. Based on this formal model, UMLbased modeling techniques are introduced. Tool support for modeling, code generation, and system execution is provided

***Towards Multiple Access in Generic Component Architectures* by M. Klein and J. Padberg (Technical University of Berlin, Germany) and F. Orejas (Universitat Politcnica de Catalunya, Barcelona, Spain)**

The paper introduces an abstract framework for the specification of components with multiple require and provide interfaces that allows the specification of multiple access to a single provide interface. This framework can be regarded as a generalization of abstract hierarchical and connector based component specification approaches. The main ideas are clarified in a sample specification, a component architecture for a web browser suite. Therefore, elementary nets are applied and are shown to be an instantiation of the abstract framework.

***Extending a Component Specification Language with Time* by Björn Metzler and Heike Wehrheim (University of Paderborn, Germany)**

In a formal approach to component specification, interfaces are usually described using pre- and post-conditions of methods or protocols. In this paper we present an approach for integrating time into a component specification language which already allows for pre/post and protocol descriptions. The specification of timing aspects is indispensable when treating components of embedded systems underlying hard real-time requirements. In order to allow for a smooth integration into the existing specification language and to ease reading and writing of interfaces, we do not extend the language with yet another formalism (for time), but instead only add a specific feature (i.e. clocks) to it. We define a semantics for this new specification language in terms of timed automata, which thus also opens the possibility of analysing interface descriptions with the UPPAAL model checker. We furthermore give timed simulation conditions and prove their soundness with respect to inclusion of timed traces, the notion of implementation in timed automata. This implementation relation can be used as a correctness criterion for interoperability and substitutability checks.



Session II: Invited Talk by Frantisek Plasil (Charles University, Prague, Czech Republic) on Hierarchical vs. Flat Component Models (joint work with Petr Hnetynka)

Component-based development (CBD) has become a commonly used technique for building software systems. There are many opinions as to what a component is. One typically agrees that it is a black-box entity with well defined interfaces and behavior, which can be reused in different contexts and without knowledge of its internal structure (i.e., without modifying its internals). However, from a design view, components - especially hierarchical ones - can be viewed as gray-box entities with the internal structure visible as a set of communicating subcomponents. Typically, the collection of the related abstractions, their semantics and the rules for component composition (creation of component architecture) is referred to as a component model. In our view, the concept of “component” has always to be interpreted in the semantics of a particular component model. Typically, the industrial component systems, such as EJB and CCM, are based on a flat component model. On the contrary, the academic component systems and models usually provide advanced features like hierarchical architectures, behavior description, dynamically updatable components, etc. Based on our experience with the SOFA and Fractal component models, we claim that a hierarchical component model provides a better support for reusability and integration of components. However, it is hard to properly balance the semantics of advanced features; this fact, in our view, hinders a widespread, industrial usage of hierarchical component models. This issue is primarily related to dynamic reconfiguration of an architecture, i.e., adding and removing components at runtime, passing references to components, etc. A simple prohibition of dynamic reconfiguration (even though adopted by some systems) would be very limiting, since dynamic changes of architecture are inherent to many component-based applications. On the other side, particularly in hierarchical component models, an arbitrary sequence of dynamic reconfiguration can lead to “uncontrolled” architectural modification, which is inherently error-prone (we call this issue the evolution gap problem). Another currently emerging paradigm is the service-oriented architecture (SOA). SOA-based systems (WebServices, etc.) are ordinary used in industry. In a high-level view, there is no difference between the SOA and CBD paradigms - both a service and component have a well defined interface, their internal structure is not visible to their environment, and they can be reused in different contexts without modification. However, in SOA, services are not nested and their composition is typically made with the granularity of each request call, frequently being data driven. Thus, because of lack of any continuity in the architecture, there is no problem with its dynamic reconfiguration. In this talk, we present an approach for handling dynamic reconfigurations of an application architecture. To avoid uncontrolled architecture modifications, the described solution is based on several reconfiguration patterns and on the introduction of the concept of a utility interface, which allows using a service provided under the SOA paradigm from a component-based system. The talk is based on our experience with non-trivial case studies built for the SOFA and Fractal component-based systems.



Session III: Component Contracts and Component Selection

***Parametric Performance Contracts: Non-Markovian Loop Modelling and an Experimental Evaluation* by Heiko Koziolk and Viktoria Firus (University of Oldenburg, Germany)**

Even with today's hardware improvements, performance problems are still common in many software systems. An approach to tackle this problem for component-based software architectures is to predict the performance during early development stages by combining performance specifications of prefabricated components. Many existing methods in the area of component-based performance prediction neglect several influence factors on the performance of a component. In this paper, we present a method to calculate the performance of component services while including influences of external services and different usages. We use stochastic regular expressions with non-Markovian loop iterations to model the abstract control flow of a software component and probability mass functions to specify the time consumption of internal and external services in a fine grain way. An experimental evaluation is reported comparing results of the approach with measurements on a component-based webserver. The evaluation yields that using measured data as inputs, our approach can predict the mean response time of a service with less than 2 percent deviation from measurements taken when executing the service in our scenarios.

***A Contract-based Approach to Specifying and Verifying Safety Critical Systems* by Wei Dong, Zhen-bang Chen and Ji Wang (National Laboratory for Parallel and Distributed Processing ChangSha, P.R.China)**

Light-weight formal method has been regarded very important for component-based safety critical system development. The paper proposes an approach which can formally specify and verify the contract of static structure, dynamic behavior and refinement of component systems based on UML 2.0 superstructure. As results, the correctness of static contract can be obtained via type checking of interfaces and connectors. Dynamic contract can be verified through determining the cooperativeness of integrated components, whose contracts are depicted with interface protocol state machines and their semantics models, namely contract automata. The refinement relation between high level component and its implementation will be guaranteed through defining the alternating simulation between contract automata of components in different hierarchies.

***Only the Best Can Make It: Optimal Component Selection* by Lars Gesellensetter and Sabine Glesner (Technical University of Berlin, Germany)**

In Component-based Software Engineering (CBSE), the construction of cost-optimal component systems is a nontrivial task. It requires not only to optimally select components and their adaptors but also to take their interplay into account. In this paper, by employing methods from the area of compiler construction and especially optimizing code generation, we present a unified approach to the construction of component systems, which allows us to first select an optimal set of components and adaptors and afterwards to create a working system by providing the necessary glue code. With our two case studies, we demonstrate that our approach is efficient and generally applicable in practical scenarios.



Session IV: Model Checking

***Local Module Checking for CTL Specifications* by Samik Basu (Iowa State University Ames, USA) and Partha S Roop and Roopak Sinha (University of Auckland, New Zealand)**

Model checking is a well known technique for the verification of finite state models using temporal logic specification. While model checking is suitable for transformational systems (also called closed systems), it is unsuitable for open systems (also known as reactive systems) where the nondeterminism in the environment must be considered during verification. Module checking is an approach for the verification of open systems which have both closed (internal) and open (environment or external) states. It has been demonstrated that the complexity of module checking branching time logic CTL is EXPTIME-complete. The approach to module checking is global and the method tries to establish that the property in question holds over all possible environments. This paper develops a local approach to CTL module checking using tableau rules. The proposed approach tries to determine a single environment under which the negation of the property is satisfied over the given module. Such a strategy, thus, leads to a local approach to module checking where we only explore states that are relevant to proving that the negation of the property can be satisfied over the given module using an appropriate witness (environment) that the algorithm also generates. While the worst case complexity of our algorithm is identical to the earlier complexity, we demonstrate that practical implementation of the proposed approach is feasible and yields much better results than the global approach.

***Specification and Generation of Environment for Model Checking of Software Components* by Pavel Parizek and Frantisek Plasil (Charles University, Prague, Czech Republic)**

Abstract. Model checking of isolated software components is inherently not possible because a component does not form a complete program with an explicit starting point. To overcome this obstacle, it is typically necessary to create an environment of the component which is the intended subject to model checking. We present our approach to automated environment generation that is based on behavior protocols; to our knowledge, this is the only environment generator designed for model checking of software components. We compare it with the approach taken in the Bandera Environment Generator tool, designed for model checking of sets of Java classes.

3 General Discussion: What can the CBSE and embedded system communities learn from each other?

Besides direct feedback and questions to the presentations, the discussions although went to more general questions. The cross-fertilisation of the members from the component-based software engineering and the embedded systems communities seemed to be a recurring topic. In particular, the two following areas are identified as very promising to continue the dialog.

Component Models: While in CBSE much research has been done to define elaborated component models (in fact, component *meta* models) many insights have been gained. For example, the role of



explicit context modelling when predicting quality attributes, the various dependencies of a component (and particularly its quality attributes) to its outside world, the need of including information on the dependencies between provided and required interfaces of a component in its specification (which leads to a component as transformer view). Many of these insights are not tied to enterprise applications, but much more would also be of use in component models for embedded systems. While superior in transferring formal methods in the commercial software development, the embedded systems community usually has a delay of several years in utilising software engineering approaches, such as the use of high-level languages, object-orientation, or right now, component orientation. This delay is certainly caused by the lack of formal underpinnings of these approaches in the software engineering community. However, particularly in the area of component models, the embedded systems community is about to replicate research present in the CBSE communities for the last years.

Modelling of quality attributes with stochastic distribution functions: While in embedded systems real-time guarantees (in form of upper bounds) have been of dominant importance due to reasons of application safety, in enterprise systems, the middleware of communication systems used to not provide any kind of real-time guarantees. As a consequence, research in quality of service of enterprise systems quality investigates stochastic distribution functions to specify and predict quality of service. The timing behaviour of many new embedded applications (such as mobiles, PDAs etc) is not bound to safety properties of the systems as it traditionally was for the embedded system community. Therefore, the use of stochastic model for quality attributes seems attractive also in the embedded systems community.

Model Checking: While research in protocol-modelling interfaces has a long history in CBSE, the use of using model checkers for proving interoperability has not been exploited to the moment. In particular, the inclusion of timing properties in interfaces and the use of the recent work on timed model checking is very promising for research in CBSE.

4 The best paper awards

As FESCA has a tradition in receiving many high quality papers, the organisers decided to introduce the FESCA best paper award to honor the achievement of the authors of the best papers among these top submissions. This year we had two exceptional papers according to the reviewers' comments. Therefore, the two best paper awards were given to:

Björn Metzler and Heike Wehrheim (University of Paderborn, Germany) for their paper *Extending a Component Specification Language with Time*. All three reviewers suggested strong acceptance. To quote from the reviews: "The paper describes the solution to an important problem, the question how time constraints for components can be expressed and checked for concrete implementations. It is a smart decision of the authors to not create yet another formalism but instead find means to express time in existing formalisms. This offers not only the advantage that specifications are more readable. Also existing theory (the semantics of the specifications) and tools (model checker)

can be extended and used for the specification variant with time, as has been demonstrated in the paper.” and “In a strife for extending their well-researched CSP-OZ formalism to specification and analysis of real-time components, the authors pursue a a technically straightforward extension of CSP-OZ by introduction of clock variables, as found in timed automata. In contrast to other extensions of the same formalism aiming at the same domain, in particular CSP-OZ-DC (extending the formalism with Duration Calculus formulae, thus integrating a full-fledged metric-time temporal logic), the current extension is technically much simpler and also considerably less expressive. This is, however, well justified by the ease of manipulation gained: there is a direct embedding of timed CSP-OZ into (potentially infinite) timed transition table, which facilitates automatic model checking with standard tools if the state space of the specification happens to be finite.”

Samik Basu (Iowa State University Ames, USA) and Partha S Roop and Roopak Sinha (University of Auckland, New Zealand) for their paper *Local Module Checking for CTL Specifications*. Again, the three reviewers all opted for strong acceptance: “This paper gives a local approach to CTL module checking by determining a single environment for which the negation of the property is satisfied. So the state-space of the module is explored locally and on-the-fly, so this technique only explores those states needed to prove the negation of the property.” and “The paper introduces a new local method for module checking based on a tableau construction. Module checking is concerned with model checking open systems, where the environment may influence the holding of a temporal logic property (here CTL). The technique tries to construct an environment which - in combination with the system - falsifies the property under consideration.” and “The main contribution of the paper is a sound and complete set of tableau rules for local CTL module checking. Though having the same worst case complexity as global module checking, the authors demonstrate that a practical implementation of their local approach yields much better results.”

5 Conclusions

Given the high quality of the presented papers, the lively discussions and the obvious cross-fertilisation of two communities, we are happy to announce that FESCA will be continued next year on ETAPS 2007. 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 hosted at King’s College.



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 societal 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, advises 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