ERCIM NEWS

European Research Consortium for Informatics and Mathematics
www.ercim.org

Number 36

January 1999

FRONT PAGE

Gerard van Oortmerssen, General Director of CWI and President of ERCIM

We live in interesting times. The way we work, communicate, trade, educate, learn, entertain ourselves and the way in which countries are governed is rapidly changing. Only five years ago few people outside the academic world knew what the internet was. Now, millions are surfing the web. Change is all around us, at an unprecedented pace. And this is only the beginning! I believe that we are on the verge of a fundamental breakthrough in human evolution.

It is evident that research is playing an important role in the process of change that we see all around us. Research creates new opportunities. At the same time, change itself generates new problems, requiring more research:

•the rapidly increasing number of users of the internet and the development towards streaming media causes congestion, so we have to develop a high bandwidth, reliable infrastructure

•the millennium bug has raised awareness of how dependent we are on all kinds of automated systems; the rapidly increasing number of embedded systems in all kinds of products is only aggravating the need for methods to check the correctness of software systems

•among the millions of people using the internet, there will be some that will try to use the internet for criminal purposes, so we have to find methods for monitoring and preventing such unwanted behaviour, and develop countermeasures and methods for secure payment systems and privacy protection

•existing legislation is not suitable to regulate and support the new kind of transactions in the global community on internet.

And these are only a few of the new problems.

Europe has always played an important role in research in information and communication technology and ERCIM now actively addresses problems such as those mentioned above among many others. ERCIM, with its potential of six thousand researchers, represents a major force on the European research scene. ERCIM members have a joint ambition: to provide Europe with the best research in information and communication technologies, contributing to the development of the information society and to the creation of wealth and well-being.

With ERCIM we are building an infrastructure to facilitate information sharing and co-operative work, thus creating a huge virtual laboratory. ERCIM members act as national nodes, representing the ICT research community in their country, in a truly European research network. ERCIM is an open network: its members co-operate with universities and industry, not just in Europe, but all over the world. ERCIM institutes co-operate in working groups and in joint projects, thus consistently increasing our consortium's internal cohesion. Moreover, ERCIM has a fellowship programme and stimulates exchange of researchers among its members.

The European Union recognizes the signs of the time and is getting ready for the challenges of the new information society with its 5th Framework Programme that is about to start. ERCIM is prepared to meet the challenge!

Gerard van Oortmerssen

SPECIAL:

CONTENTS

ERCIM will celebrate its 10th anniversary with a two days event in Amsterdam, 4-5 November 1999. See announcement on page 3.

Next Issue:

Networking Technologies

CALL FOR PROPOSALS

Prospective Reports on Research in Information and Communi-cation Science and Technology in Europe

On the occasion of its 10th anniversary in 1999, ERCIM wishes to contribute to develop a long-term vision on the future of Information and Communication Science and Technology (ICST) research in Europe and related fields, within, say, a five-year horizon. This call is aimed at encouraging European computer scientists and applied mathematicians are invited to write prospective reports. These reports should emphasise such issues as scientific challenges in computer science and applied mathematics, development of interactions with other sciences, new prospects of applications, job-creating potentialities, identification of centres of excellence in Europe and world-wide.

ERCIM Working Groups and scientists from ERCIMinstitutes are most welcome to submit, but this call for proposals is open to any European scientist.

Possible topics considered important by ERCIM for these prospective reports include the main scientific and technological challenges of ICST (such as high speed networking, distributed and mobile computing, software engineering and certification, computational linguistics, co-operative work, intelligent information retrieval, user interfaces for all, computer vision and image processing, multimedia and virtual reality, optimisation, control and systems theory, large-scale simulations, etc), the economical and social impact of these technologies (such as applications of ICST in medicine, telecommunications, transportation systems, environmental modelling, virtual museums, education, commerce, finance, etc), and also questions concerning the ICST research community and its future (such as: relations of computer science with other scientific disciplines, evolution of computer science curriculae, etc).

Suggested Formats

1. A possible format consists in organising a workshop bringing together senior scientists from academia or applied research institutes and industrial companies (possibly also from health care sector, governmental agencies, etc), coming from at least four European countries. A prospective report of 10 to 15 pages will be written up after the workshop. If the proposal is selected, a lump sum of 6 kECU will be granted by ERCIM to cover the travels of the participants and the cost of logistics associated with the meeting. An extra sum of 2 kECU will be allocated to the scientist(s) in charge of writing the report, after submission of an extended abstract and of a summary of the workshop discussions.

2. An alternative format may consist in the sole writing-up of a prospective report, by a senior European scientist (or a small number of them). If the proposal is selected, a travel budget (up to 2kECU) will be granted by ERCIM. An extra sum of 2 kECU will be allocated after submission of an extended abstract of the prospective reports.

Proposals

The proposals should contain:

- a one-page description of the field covered by the expected prospective report
- the CVs of the author(s) in charge of writing the prospective reports, and a brief description of their motivation and achievements with respect to the proposed prospective report
- for the workshop format, information about the workshop (location and date, hosting organisation, list of participants...)
- for the report format, a proposed travel budget (up to 2 kECU)
- the complete address of the person in charge of the proposal (with phone and fax numbers, and E-mail address).

Deadlines

- proposals are to be sent to the ERCIM Office before 31 January 1999
- workshops are to be held before 30 June 1999
- extended abstracts are to be sent to the ERCIM Office before 15 July 1999
- final reports are to be sent to the ERCIM Office by 30 August 1999.

Selection Committee

The selection process for selecting the proposals will be conducted by a committee of ERCIM institutes' Directors: Morten King (DANIT), Bernard Larrouturou (INRIA), Stelios Orphanoudakis (FORTH), Peter Inzelt (SZTAKI) and Albert Westwood (RAL).

Dissemination of the reports

The prospective reports, outcome of the present Call, will be published by ERCIM. This document will be available from the ERCIM Office and advertised in the 10th anniversary issue of the ERCIM News. The reports will also be available on the web site of ERCIM.

The best report(s) may also be presented by the authors in the framework of the events scheduled for the celebration of ERCIM's 10th anniversary 4-5 November 1999 in Amsterdam, for which ERCIM strives to attract a broad participation from within ERCIM as well as from industry, European Commission and national government bodies. A decision on the reports to be presented during the events will be based upon the evaluation of the selection committee. ■

Please contact:

Bernard Larrouturou – ERCIM Manager
Tel: +33 1 3963 5303
E-mail: office@ercim.org

Matthias Grossglauser Cor Baayen Award Winner 1998

Swiss born Matthias Grossglauser is the winner of the 1998 Cor Baayen Award competition. The 5000 ECU award was presented during the ERCIM Meetings on the 5th of November in Chilton, UK by the ERCIM President, Gerard van Oortmerssen. The Cor Baayen award was created to honour the first ERCIM President and is given each year to the most promising young researcher working in one of the ERCIM institutes.

ERCIM President Gerard van Oortmerssen (right) presenting the Cor Baayen Award to Matthias Grossglauser.

The work for which Grossglauser – an EPFL-graduate – received the award was carried out at INRIA Sophia Antipolis under the guidance of Jean Bolot where he was a member of the RODEO team. He defended his thesis in spring 1998 on the topic 'Control of Network Resources over Multiple Time-Scales'. The topic is clearly of major importance, with regards to the ever increasing role played by networks at the present time. He is a researcher with a sizable publication list and his work is of outstanding quality. His research concentrates on fundamental design principles for network services, protocols, and control mechanisms; resource allocation and sharing; performance measurement and analysis. Grossglauser recently joined AT&T Laboratories in the US. An article on his work will be published in the April 1999 issue of ERCIM News.

The Cor Baayen award, up to now restricted to researchers working in one of the fourteen ERCIM institutes, will be open from next year on to any young researcher having completed his/her PhD-thesis in one of the fourteen 'ERCIM countries': Czech Republic, Denmark, Finland, France, Germany, Greece, Hungary, Italy, Norway, Slovakia, Sweden, Switzerland, The Netherlands and the UK. The maximal two candidates per country have to be nominated by the corresponding ERCIM member institute. For details, see http://www.ercim.org/activity/cor-baayen.html ■

Please contact:

Frans Snijders – CWI
Cor Baayen Award Co-ordinator
Tel: +31 20 592 4171/4009
E-mail: Frans.Snijders@cwi.nl

ERCIM 10th Anniversary Event

Amsterdam, 4-5 November 1999

ERCIM will celebrate its 10th anniversary with a two days event in Amsterdam, 4-5 November 1999. The first day will be an internal event for ERCIM-member personnel only, while the second day will be dedicated to Information Technolgy users in industry.

ERCIM – a virtual laboratory for IT research in Europe, Amsterdam, Thursday 4 November 1999

Under this slogan scientists of the ERCIM institutes will be given the opportunity to present their ideas on matters that are closely related to IT research. It is not research itself that should be targeted with these presentations but rather the issues that come up on a meta-level. To give some examples of the topics that might be tackled, we mention: 'The future of traditional scientific journals in the Internet Age 'Should Computer Science and Mathematics be kept separate?', 'Is freeware (cooperative development of software) a viable alternative to the traditional model?', 'Why is software less reliable than a washing machine?', 'Why Framework Programs don't work', 'Alternatives to traditional scientific output metrics', 'Are European CS-curricula too theoretical?' Note that this is not an exhaustive list but just an indication. The presented ideas should be provocative and express opinions of individuals, not necessarily shared by their home institutes.

ERCIM scientists interested to contribute to this event are invited to contact their Executive Committee representative.

ERCIM – leveraging European R&D for Business and Society, Amsterdam, Friday 5 November 1999

With this event ERCIM will provide a forum for the presentation, discussion and exchange of ideas between people from industry, R&D and government. Representatives from European industry, the European Commission and national governmental bodies will be invited to meet with ERCIM scientists, attend presentations of top research projects and listen to visionary speeches given by leading people from European industry and research organizations.

Please contact:

ERCIM office
Tel +33 1 3963 5303
E-mail: office@ercim.org

8th DELOS Workshop on User Interfaces in Digital Libraries

by Preben Hansen

The 8th DELOS Workshop on User Interfaces in Digital Libraries was held in Stockholm, Sweden, 21-23 October 1998. The DELOS Working Group is an action of the ERCIM Digital Library Initiative (http://www.area.pi.cnr.it/ErcimDL/). During the workshop, a 'mini-workshop' was held, together with participants from the 4th ERCIM User Interfaces for All Workshop, another ERCIM Working Group, that was held from 19-21 October 1998. 21 participants, including guest speakers attended the workshop. Eleven presentations were made during the workshop.

A Digital Library is the integration of several different components and will include a range of content and services, as well as a large and diverse group of users. It is important to develop an understanding of the overall tasks and interactions users are engaged in when entering a Digital Library. We interact constantly with our environment through different communication mechanisms and processes. Information seeking and retrieval in Digital Libraries is but a special case of such a process. Analysis and evaluation of user, systems and interactions are needed to successfully build future Digital Libraries.

Presentations

The workshop started with a 'Challenge paper' (Preben Hansen and Jussi Karlgren, SICS). The paper summarized some important research issues raised 25 years ago related to information retrieval (IR) and user interfaces (UI) and its relevance for the workshop. The authors found that some of the questions raised then were still valid today, such as the characteristics of the user, the task, the information content and medium, the computer and IR techniques and the role of evaluation and feedback in the redesign cycle, among others. However, there has also emerged new research areas such as multimedia content, multimodal interaction, multilingual information and users and distributed systems and collections.

Our guest speaker, Professor Nicholas Belkin, Rutgers University, provided a 'road-map' on important issues for Digital Libraries in his paper 'Understanding and supporting Multiple Information Seeking Behaviours in a Single Interface Framework'. First, Belkin presented a definition of a Digital Library and what functions need to be supported in such a framework. Furthermore, Nick Belkin described his and his group's work within the third TIPSTER research program. The goal of the project is to identify and classify different information seeking strategies (ISSs), characterize sequential structures of ISSs, identify specific combinations of IR techniques appropriate for different ISSs, and construct and evaluate system which adapts to support different ISSs in the course of a single information seeking episode.

Constantine Stephanidis, ICS-FORTH, our second guest speaker, raised some critical issues for interaction design in digital libraries in the light of HCI and Digital Libraries. Among the main issues and challenges mentioned were diverse user groups, variety in the context of use, and technological proliferation. The author also proposed a way to deal with the design of Digital Libraries containing three phases: Design processes and techniques, Implementation, and Evaluation.

The paper presentations covered areas such as multilingual aspects (Nuño Miguel Antunes Freire, INESC and Aarno Jarno Tenni, VTT); evaluation of information systems (Silvana Mangiaracina, CNR and Demosthenes Akoumianakis, ICS-FORTH); agent-based system of semantical information retrieval (Kuldar Taveter, VTT); visualization (Francesca Costabile, Bari University); implementation of user interfaces (Donatella Castelli, CNR and László Kovács, SZTAKI) and finally, genres and clustering (by Johan Dewe and Niklas Wolkert, Netsolutions AB).

Discussion summary

Some important issues discussed during the workshop:

- information seeking and retrieval as embedded activities within Digital Libraries
- techniques and methods to analyse, and evaluate different systems as well as different users, their behaviour, tasks and the ideas behind the systems developed
- support interactions with information, such as texts and multimedia and access to multilingual information in information seeking activities
- future Digital Libraries will encompass alternative modalities for representations of information seeking activities.

The papers from the workshop will be published as workshop proceedings at http://www.ercim.org/publications/ws-proceedings/.

■

Please contact:

Preben Hansen – SICS
Tel: +46 8 633 1554
E-mail: preben@sics.se

4th ERCIM Workshop on User Interfaces for All

by Constantine Stephanidis

The 4th workshop of the ERCIM Working Group on User Interfaces for All (UI4ALL) took place in Sweden 19-21 October 1998, in the rather arresting settings of the Stockholm urban archipelago in the former Långholmen jailhouse. The local organiser was Dr. Annika Waern, of the Swedish Institute of Computer Science (SICS). In addition to the wide ranging topics addressed annually, this year's workshop focused on the special theme 'Towards an Accessible Web', and attracted considerable interest, within, but also beyond Europe.

The Workshop featured two invited speakers, both of them working in the field of Web accessibility and affiliated with the activities of W3C-WAI (Web Accessibility Initiative): Dr Daniel Dardailler, Project Manager of W3C-WAI, and Dr Michael Paciello, from the Yuri Rubinsky Insight Foundation, Canada, and the Web Able Solutions, USA.

The paper presentations and the discussions during the workshop covered a variety of related topics, including: Design Methodologies for Universal Access, Extending the Browser Metaphor, Adaptivity and Adaptiveness, WWW Browsers for All, Design Principles and Guidelines, and Information Filtering and Presentation.

The topics addressed current and on-going activities in the broader context of universal in the emerging Information Society. Particular reference was made to the aims and current achievements of the W3C-WAI, as well as on the results of the ACTS-AC042 AVANTI project, which, amongst other things, involved the development of an adaptable and adaptive Web browser (four ERCIM member organisations, CNR-IROE, GMD-FIT, VTT and FORTH-ICS have participated in the AVANTI consortium).

Following a stringent peer review process, the Workshop's proceedings have included three main categories of articles accepted for publication: (a) 7 long papers, (b) 4 short papers, and (c) 4 position papers. Additionally 3 posters were presented during the interactive poster session at the Workshop. The proceedings of the Workshop are electronically available via the Web site of the ERCIM Working Group on 'User Interfaces for All' at: http://www.ics. forth.gr/ercim-wg-ui4all

In the morning of 21st of October 1998, the Annual General Meeting of the ERCIM UI4ALL Working Group was held at the same location. Following a review of recent progress in the field, the group focused on the opportunities for the submission of project proposals in the 5th Framework Programme, the planning of the next WG Annual Meetings, the prospect of cooperation with the DELOS WG for drafting a joint document (White Paper), and mechanisms for more systematic collaboration between the two ERCIM Working Groups.

The WG has confirmed the dates for the 5th Annual Workshop to be held in Germany (GMD), 3-5 November 1999; local organisers will be Alfred Kobsa and Michael Pieper. The first call for the 5th Annual Workshop will be out in January 1999, the second call in April 1999, and the deadline for paper submission will be 1st of July 1999. The WG has also decided to have its 6th Annual Workshop in Florence, Italy (CNR-IROE), in October 2000, and the local organiser will be Pier Luigi Emiliani. ■

Please contact:

Constantine Stephanidis – FORTH
Tel: +30 81 391741
E-mail: cs@ics.forth.gr

ERCIM DELOS - User Interfaces for All Joint Workshop

by Constantine Stephanidis and Preben Hansen

On the occasion of the co-location of the 4th Annual ERCIM Workshop on 'User Interfaces for All', and the 8th ERCIM DELOS Workshop on 'User Interfaces for Digital Libraries', which took place in Stockholm, Sweden, 19-21 October and 21-23 October 1998 respectively, a joint meeting (mini-workshop) was organised. The objective of this meeting was to discuss and elaborate an R&D agenda for HCI activities in the field of Digital Libraries.

The main issues that were discussed during the meeting include the characterisation of the current situation concerning the area of interaction design for Digital Libraries (particularly in the context of R&D activities of the two working groups), the opportunities for the submission of proposals for joint project work in the context of the 5th Framework Programme of the European Commission, and the drafting of a White Paper to articulate the vision and scope of the common R&D directions.

During this workshop, critical R&D themes of common interest emerged pertaining to the contributions from information retrieval, co-operation and collaboration, interaction metaphors, visualisation, ubiquitous access, customisation and individualisation, interoperability, etc, to facilitate both the informational and situational (eg social) aspects of Digital Library systems. In addition, non-technological aspects such as ethics, intellectual property rights and the role of the end-users were deemed as crucial elements of success in constructing and deploying Digital Library systems.

In the course of this workshop, the opportunity to compile a White Paper to reflect the common ground and provide a joint R&D roadmap for future actions in this area was received very favourably. The White Paper is expected to give an account of the state of the art internationally, to identify existing gaps in current R&D efforts, and to synthesise the range of critical issues that need to be addressed in the context of new co-operative R&D activities. ■

Please contact:

Constantine Stephanidis – FORTH
Tel: +30 81 391741
E-mail: cs@ics.forth.gr

Preben Hansen – SICS
Tel: + 46 8 6331554
E-mail: preben@sics.se

DELOS Workshop on Emerging Technologies in the Digital Libraries Domain

by Carol Peters

The objective of the DELOS Working Group, funded by the ESPRIT Long Term Research Programme, is to promote research into the further development of digital library technologies. Since 1996, DELOS has organised Workshops on DL-related research topics (see this number for a report on the 8th DELOS Workshop on User Interfaces), has been responsible for initiating a series of European Conferences on Digital Library Research and Advanced Technology (ECDL'99 will be held next September in Paris) and has partially financed the setting up of the ERCIM Technical Reference Digital Library (ETRDL). In addition, DELOS has sponsored five European-US collaborative working groups.

A digital library is the integration of multiple components which do not initially fit together in a seamless fashion for a number of reasons. Firstly, the necessary components come from a background of different communities and, secondly, they should enable new functions which were not under consideration when the individual single components were first designed and implemented. This means that the realisation of large-scale globally distributed digital libraries depends as much on collaborative effort as it does on the development of new technologies in order to develop systems which truly integrate their components. A high level of collaboration is required both across disciplines and across geographical boundaries. For this reason, DELOS in collaboration with the US National Research Foundation decided to set up a group of EU-US working groups with the mandate to jointly explore technical, social and economic issues and plan common research agendas with respect to a set of key DL research areas in which international cooperation was considered to be of particular importance. The Working Groups addressed the following DL-related research areas:

- interoperability between digital library systems
- metadata
- intellectual property rights and economic issues
- resource indexing and discovery in a globally distributed digital library
- multilingual information access.

Each group studied the state-of-the-art and current trends in their area and produced a set of recommendations and priorities for future R&D activities.

Workshop participants.

Panel Discussion: (from left to right) Simon Bensasson, EC – Long Term Research; Stephen Griffin, US National Science Foundation; Dennis Tsichritzis, GMD; Roberto Cencioni, EC - Telematics Language Engineering; Shigeo Sugimoto, University of Library and Information Science, Japan; Bernard Smith, EC – Telematics Information Engineering.

The results of these studies were presented at a Workshop, held in Brussels on 12 October, 1998, and sponsored by DELOS and by ERCIM as part of the ERCIM Digital Library Initiative. During the morning session the European and US Coordinators of the Working Groups (Costantino Thanos, IEI-CNR, and Dan Atkins, University of Michigan) described the objectives of the EU-NSF collaboration and their vision for the future of Digital Libraries to an invited audience composed mainly of European research coordinators, funding officials and heads of national research programmes, plus some representatives from leading industries interested in aspects of Digital Library research and applications. European and US leaders of the working groups (Christos Nikolaou, University of Crete; Carl Lagoze, Cornell University; Hans-Jörg Schek, ETH-Zurich; Thomas Baker, GMD, Bonn; Judith Klavans, Columbia University, US) then outlined the main proposals for each area of activity. The session ended with a summary of the global recommendations of the groups for the efficient and effective development of the next generation of digital library systems presented by Peter Schäuble, ETH-Zurich and Alan Smeaton, City University, Dublin.

In the afternoon, a panel discussion, chaired by Dennis Tsichritzis, GMD and former president of ERCIM, gave Programme Officers of the European Commission and the US National Science Foundation and a representative of the Japanese DL community the opportunity to present their own opinions and the views of their agencies with respect to developments in the digital library area and discuss future scenarios for cooperative actions.

The first results of the joint EU-NSF Working Groups, summarised as 'An International Research Agenda for Digital Libraries', have been published by ERCIM. Final reports by the five Groups will be available early 1999. For an on-line version of the summary report and for further information on the activities of the DELOS Working Group, see http://www.iei.pi.cnr.it/DELOS/ ■

Please contact:

Costantino Thanos – IEI-CNR
DELOS Coordinator
Tel: +39 050 593 492
E-mail: thanos@iei.pi.cnr.it

4th ERCIM Environmental Modelling Group Workshop

by Thomas Lux

The fourth workshop of the ERCIM Environmental Modelling Group on Environmental Models and Computational Methods was held in Heraklion, Crete, Greece on 16-17 November 1998. The workshop was hosted by IACM FORTH, the Institute of Applied and Computational Mathematics of the Foundation for Research and Technology Hellas under the chairmanship of Nikolaos A. Kampanis (IACM FORTH).

The lectures and discussions at the workshop focussed especially on possible improvements concerning the degree that physical phenomena are interpreted through the governing equations. This may be accomplished by

exploiting, through the exchange of experience among the participants, the robustness, efficiency and range of applicability of modern computational methods.

Lectures presenting recent results of experimental methods for the analysis of atmospheric dynamics, the modelling and simulation of coastal water pollution, and the investigation of aquatic ecosystems have considerably enlarged the field of environmental domains studied within the frame of the working group.

A special item of the workshop program was a round table discussion about the further activities of the ERCIM Working Group Environmental Modelling. Possible places and dates of the next workshop have been raised. Moreover, topics for future joint project proposals within the Fifth Framework Programme

The participants of the 4th ERCIM Environmental Modelling Workshop.

advancing the dialog among researchers working in the field of environmental modelling on the issue of computational methods. Modern high-performance computers provide a powerful base for improving existing and developing new computational techniques for the efficient solution of complex models used for the numerical simulation of various environmental processes. Further, discussions has been initiated

on the efficiency and accuracy of current environmental models, as well as

of the European Union have been presented. The program was wound up by a common dinner in a typical Crete restaurant. Detailed information about the workshop program and the participants can be found at http://gorgona.iacm.forth.gr/~flouri/ERCIM/index.html. ■

Please contact:

Achim Sydow, Working Group Chairman or Thomas Lux – GMD
Tel: +49 30 6392 1820
E-mail: lux@first.gmd.de

Programming Language Technologies

by Neil D. Jones

Mankind is the species that uses and makes tools; but it is just as uniquely the species that uses and makes languages. Once computers greatly extended our tools' reach, effective power, and degree of automation, it was inevitable that sophisticated control and interaction mechanisms have become indispensable for controlling today's advanced and automatic tools. The language skill enters just here: we are able not only to communicate, but also to design special-purpose command and communication protocols, and to design algorithms that manipulate programs as data objects. Further, it is increasingly routine to use the computer to verify (by symbol manipulation!) that programs achieve their purposes correctly and without failure, deadlock, or issuing unsafe commands to the tools they steer.

The focus of this ERCIM News Special Theme is primarily on the program itself, as an object of study or subject of manipulation; or on a programming language in which programs may be written. Thus this Working Group takes a somewhat different view than other (admittedly closely related) Computer Science activities that also yield programs as outputs, or use programs as tools: Software Engineering, Formal Methods, Human-Computer Interaction, Computer-Supported Cooperative Work etc. The main point is that sometimes *programming language can be a tool* for solving a class of problems. It thus behooves us better to understand our artificial languages, just as deeper understanding of physical tools assisted the industrial revolution.

The contributions to this issue witness widespread language-related activity within ERCIM. They encompass a wide range including:

- a language as a solution, establishing a framework or viewpoint, or giving users new capabilities
- tools manipulating languages: compilers, interpreters, program analysers, etc
- ways to solve problems involving programs, in particular legacy code and the notorious Year 2000 problem
- semantics: not as an ivory-tower end in itself, but as a crystallisation of the essence of a particular programming language that can be used as a basis for programming environments
- implementation perspectives: what *can* be done (engineering), *limits* (pragmatic or theoretical studies of complexity), how to assess program complexity, or to understand programs
- ways to generate programs automatically: essential in the long run, if human bottlenecks are to be overcome.

Invited papers: the first two are from outside ERCIM: the Bandera Project uses abstract interpretation, partial evaluation and model checking to certify Ada program correctness; and the ETI platform allows wide-range experimentation with prototype software tools.

The next group concerns *software re-engineering:* how to deal with the

CONTENTS

existing masses of old, undisciplined but indispensable programs? A semantics-based and successful Year 2000 conversion tool is described. Further works describe systematic, serious approaches to dealing with legacy code in general.

After initial disappointments based on unrealistic expectations, steady progress has occurred in *automatic program analysis and verification:* the topic of the next group of papers, all of which concern validation of programs in real-world rather than academic languages.

Recent years have seen increasing activity in automatic *program generation*, here witnessed by three papers based respectively on partial evaluation, code skeletons, and program transformation. Here, a good semantic basis has had clear practical consequences. Further, the heavy task of *compiler development* has been eased by increasingly sophisticated, flexible, modular, and user-friendly high-level tools and languages.

The next group of papers concerns *Java and other object-oriented* languages. These pragmatically successful languages are becoming better understood in both practice and theory, resulting in more sophisticated tools. The next three papers witness a similar advance in the fields of *distributed systems and mobile computing:* subjects which are taking clearer form, partly aided by appropriate programming language formalisms.

Finally, there is still scope for new programming languages, and the last group of papers describes several: a now well-established object-oriented version of ML; a graphical environment for parallel programming; and the use of rewrite rules to link practical programming with mathematics as known from algebra.

■

Please contact:

Neil D. Jones – DIKU/DANIT
Programming Language Technologies Working Group chairman
Tel: +45 35 32 14 10
E-mail: neil@diku.dk

Bandera: Tools for Automated Reasoning about Software System Behaviour

by Matthew Dwyer, John Hatcliff, and David Schmidt

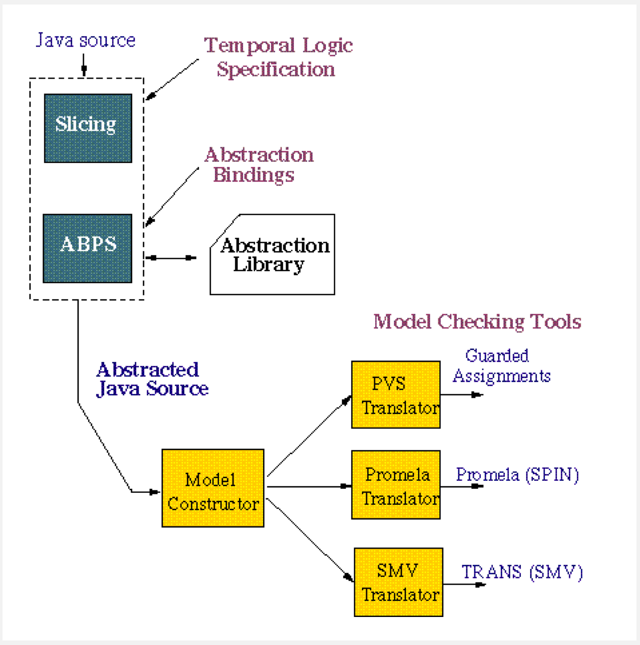
The Bandera Project aims to develop techniques and tools for automated reasoning about software system behavior, and to apply these tools to construct high-confidence mission-critical software. Automated reasoning is achieved by (1) mechanically creating high-level models of software systems using abstract interpretation and partial evaluation technologies, and then (2) employing model-checking techniques to automatically verify that software specifications are satisfied by the model. This project is a collaborative effort between the Laboratory for Specification, Analysis and Transformation of Software (SANTOS) at Kansas State University and researchers at the Universities of Hawaii and Massachusetts. Work in SANTOS is supported in part by grants from the US National Aeronautics and Space Administration (NASA), the Defense Advanced Research Projects Agency (DARPA) and the National Science Foundation (NSF).

Modern mission-critical software systems tend to be highly complex and concurrent, and they often have stringent correctness requirements. Pre-deployment reasoning about system behavior is crucial in application areas such as avionics, industrial, health care, military command-and-control where the cost of failure in the field is extremely high. Unfortunately, the inherent size and complexity of such systems prohibit classical validation techniques, such as testing, from providing high levels of assurance of reliability and correctness.

Subtle timing-related defects in concurrent and embedded systems are very difficult to reveal through testing. To do so would generally require software testers to exercise all feasible execution paths and all possible interactions between software components. In modern systems, this is virtually impossible, and many deployed systems fail when real-world use leads to an execution path that was not foreseen by the software designers. Better verification techniques are sorely needed.

The Bandera tools allow software designers and quality assurance personnel to state properties about software source code that must hold along all execution paths. For example, in a priority-based concurrent system developers might want to assure that: (1) when two components of different priority levels attempt to access a shared resource the higher-priority components gets access first; (2) when a component attempts to access a resource it eventually succeeds. Ideally, the tools should automatically explore all execution paths and check to see if the given properties hold. With some properties this can be done, but in most cases the size of the system and the complexity of the specification makes this infeasible. In these cases, the user guides the tools in the construction of a smaller abstract model of the software system, and then the tools can automatically check to see if a given specification holds in the model. If the property holds in the model and if the model safely approximates the software system's behavior, this guarantees that the property also holds in the original software system. If the property does not hold, the tools will generate a counter-example —- a trace in the model system that violates the given property. By appropriately interpreting the counterexample, one can locate the source of the offending defect in the system being modeled.

In the process outlined above, safe and effective model creation is crucial for the approach to be successful. To make automatic checking tractable, the model must discard information about the program that is irrelevant to the property being verified. However, it must retain enough structure to reason about relevant execution paths. The Bandera tools use abstract interpretation, a rigorous semantics-based methodology for constructing static analyses of programs, to form safe abstractions of software, and partial evaluation and slicing techniques to build compact models. The figure gives the architecture of the tool suite for reasoning about Java programs.



Architecture of the tool suite for reasoning about Java programs.

Given software component source code S and a property to be verified P, a slicing tool then cuts away portions of S that are irrelevant for verifying P. The user then selects abstract interpretation definitions to be used in abstracting the remaining program components. These definitions can be drawn from a library of common abstractions or they can be defined from scratch. Once the abstractions are specified, abstraction-based partial evaluation (ABPE) creates an abstracted and specialized version of the source program. The transition system generator compiles the abstract program to one of several existing model-checking tool input languages. If verification fails, counter-examples produced by the model-checking tools are rendered in terms of the original source program.

We have applied this methodology to several software systems written in Ada. We have validated correctness properties of a programming framework that supports parallelization of worklist algorithms; this framework has been used to implement a variety of scientific computing applications. We have also performed unit-level verification of generic stack, queue, and priority-queue implementations and demonstrated the ability of model checking to detect realistic implementation defects in such systems. The lessons learned from these experiences have provided important validation of and feedback to our ongoing design and implementation of Java model construction tools. We are working with the automated software engineering group at the NASA Ames Research Center to incorporate these tools into their avionics software workbench.

More information about this project is available at:
http://www.cis.ksu.edu/~santos

■

Please contact:

Matthew Dwyer, John Hatcliff, and David Schmidt – SANTOS Laboratory, Department of Computing and Information Sciences, Kansas State University
E-mail: {dwyer, hatcliff, schmidt}@cis.ksu.edu

ETI: An Online Service for Tool Co-ordination

by Bernhard Steffen, Tiziana Margaria, and Volker Braun

The Electronic Tool Integration platform (ETI) associated to the International Journal on Software Tools for Technology Transfer (STTT) is designed for the interactive experimentation with and co-ordination of heterogeneous tools. ETI users are assisted by an advanced, personalised Online Service guiding experimentation, co-ordination and simple browsing of the available tool repository according to their degree of experience. In particular, this allows even newcomers to orient themselves in the wealth of existing tools and to identify the most appropriate collection of tools to solve their own application-specific tasks.

ETI contains and manages a heterogeneous wealth of information, functionalities and data. Currently this comprises verification tools for real time systems and model checkers. The integration of programming language tools like type checkers, optimisers and code generators is on the way. The ETI Service can be accessed via its homepage, http://eti.cs.uni-dortmund.de. From there, users can:

- access online information on the tools via hyperlinks to each tool’s home site
- access online a stand-alone version of each tool, centrally located at the ETI service sites
- access the ETI repository of integrated tools. It contains a collection of functionalities offered by the individual tools, classified for ease of retrieval according to behavioural and interfacing criteria
- experience tools and functionalities, by (a) running the (stand-alone or integrated) tools on libraries of examples, case studies, and benchmarks made available on the ETI platform, testing and running single tool

functionalities, capturing specific features offered by different integrated tools on the same examples from within a uniform graphical user interface provided by ETI, (c) constructing own application-specific heterogeneous tools through combination of functionalities coming from different tools within the ETI platform, (d) loosely specifying co-ordination tasks, which can be then automatically completed to executable tool sequences by means of ETI's co-ordination support; this, in particular, takes care of data format incompatibilities

- experiment with own sets of data, to be deployed in user-specific, protected home areas.

The wealth of input/output formats makes correct tool combination extremely difficult. ETI therefore provides co-ordination support in order to ease usability: based on its interfacing layer, which organizes a growing library of type transformers, whenever possible, type-incorrect tool sequences are automatically completed to directly executable ones. This mechanism, which is based on model synthesis for temporal logics, is hidden from newcomers. Experts, however, are able to investigate the full potential for type completion in order to flexibly exploit the entire tool repository.

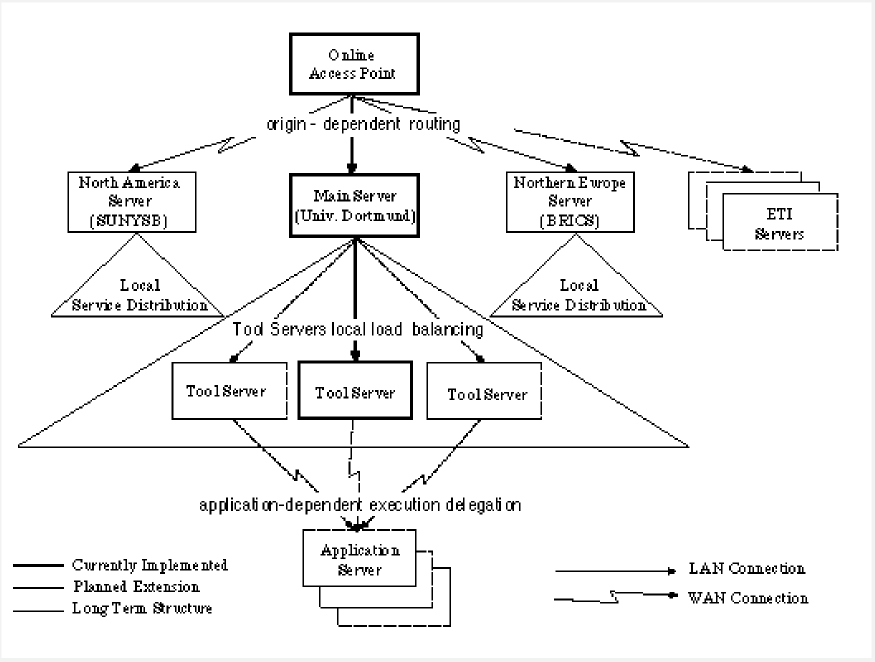
In addition, ETI provides high-level task specification languages, graphical support for specifications and user interaction, as well as prototype animation. Together this eases the access and use of the functionalities offered by different tools, even if implemented in different languages of different programming paradigms (functional, imperative, object-oriented) and running on different platforms. Together with the loose specification of single functionalities (simply in terms of desired properties), this allows even newcomers to develop and test complex tool combinations in a comfortable, intuitive manner.

The figure shows the global multi-tier architecture underlying the ETI service: via a unique entry point, visitors are routed to the most appropriate ETI Server which manages their session. The actual tool execution is a matter of the Tool Servers, which may themselves delegate tool execution to an Application Server. Typically this is the case if the requested tool does not run under UNIX or LINUX.

The ETI Online Service plays a public service role, giving users the possibility of direct, hands-on, experience with a wealth of available tools and functionalities. This also includes features like the ETI Online Forum, where users may eg, propose case studies, and report on their experiences. The service is intended to develop into a collaborative, independent tool presentation and evaluation site: users are invited to report on their experience with the integrated tools in the context of the service as a:

- directory for possible tools and algorithms satisfying totally or partially their needs
- (vendor- and producer-) independent test site for trying and comparing alternative products and solutions without any installation overhead
- quality assessment site for the published tools, which are refereed according to requirements like originality, usability, installability, stability, performance, design
- independent benchmarking site for performance on a growing basis of problems and case studies.

This should stimulate the communication between tool builders and tool users as well as between academia and industrial practice, supporting the transfer of tool-related technology. In fact, we are optimistic that the typical hesitation to try out new technologies can be overcome since serious hurdles, like installation of the tools, getting acquainted with new user interfaces, lack of direct comparability of the results and of performances, are eliminated. Moreover, the intended collaborative effort of the ETI user community to provide easily accessible information about fair, application-specific evaluations of various competing tools on the basis of predefined benchmarks, will be of substantial help for everybody in need of tool support. ■


Online Access Point
origin - dependent routing
North America Server (SUNYSB)
Main Server (Univ. Dortmund)
Northern Europe Server (BRICS)
ETI Servers
Local Service Distribution
Local Service Distribution
Tool Servers local load balancing
Tool Server
Tool Server
Tool Server
application-dependent execution delegation
Application Server
Currently Implemented
Planned Extension
Long Term Structure
LAN Connection
WAN Connection

Distributed Architecture of the ETI Service.

Please contact:

Volker Braun – Universität Dortmund
Tel: +49 231 755 5806
E-mail: eti@eti.cs.uni-dortmund.de

AnnoDomini: From Type Theory to a Year 2000 Conversion Tool

by Peter Harry Eidorff, Fritz Henglein, Christian Mossin, Henning Niss, Morten Heine Sørensen and Mads Tofte

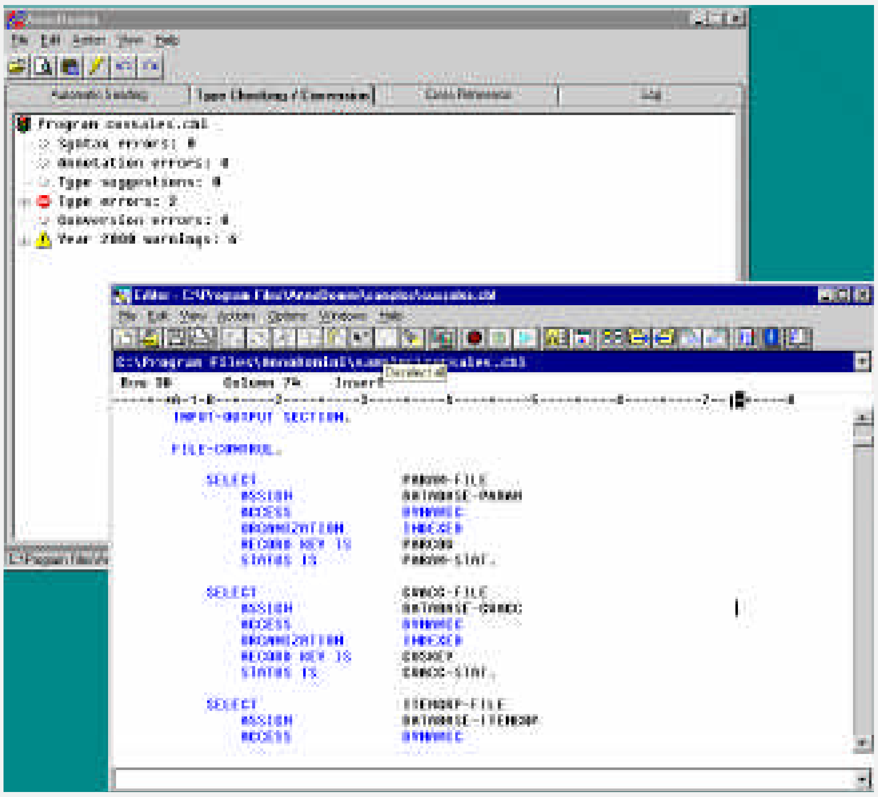
AnnoDomini is a commercially available source-to-source conversion tool for making COBOL programs Year 2000 compliant. It was developed in the last two years by a group at DIKU (part of the ERCIM partner DANIT) and grew directly out of research in the theory of programming languages; it uses type-based specification, analysis, and transformation. These are combined into an integrated software reengineering tool and method for finding and fixing Year 2000 problems. AnnoDomini's primary goals have been flexibility, completeness, correctness, and a high degree of safe automation.

The Year 2000 (Y2K) Problem refers to the inability of software and hardware systems to process dates in the 21st century correctly. The problem arises from representing calendar years by their last two digits and thus restricting the range of representable years to 1900-1999 A.D. Making programs Y2K compliant involves date field expansion or masking. Expansion refers to expanding unsafe 2-digit years to 4-digit years. Masking denotes a variety of methods for extending 2-byte year representations into the 21st Century. In windowing, for example, a pivot year determines whether a two-digit year belongs to the 20th or the 21st century. For example, with pivot 50, 59 represents 1959 A.D. and 41 represents 2041 A.D.

AnnoDomini (registered trademark of Hafnium ApS) is a method and source-to-source conversion tool, developed at DIKU, for making COBOL programs Year 2000 compliant. It runs on Windows NT 4.0 and Windows 9X, and is commercially available from Computer Generated Solutions, Inc. (an IBM business partner) – see http://www.cgsinc.com and http://www.hafnium.com. The converted programs do not require special compiler support, but compile and execute in their existing operating environment.

program's variables, eg file descriptions, are annotated with types that specify where years occur in them, if at all. AnnoDomini also has support for automatic seeding.

In the second phase AnnoDomini propagates the seeding information to other data by type inference. For example, if the program assigns the contents of one variable to another, AnnoDomini infers that they must have



```
Program cassales.cbl
  Syntax errors: 0
  Annotation errors: 0
  Type suggestions: 0
  Type errors: 2
  Conversion errors: 0
  Year 2000 warnings: 6
```

```
       INPUT-OUTPUT SECTION.

       FILE-CONTROL.

           SELECT                      PARAM-FILE
               ASSIGN                  DATABASE-PARAM
               ACCESS                  DYNAMIC
               ORGANIZATION            INDEXED
               RECORD KEY IS           PARCOD
               STATUS IS               PARAM-STAT.

           SELECT                      CANCC-FILE
               ASSIGN                  DATABASE-CANCC
               ACCESS                  DYNAMIC
               ORGANIZATION            INDEXED
               RECORD KEY IS           CCSKEY
               STATUS IS               CANCC-STAT.

           SELECT                      ITEMORP-FILE
               ASSIGN                  DATABASE-ITEMORP
               ACCESS                  DYNAMIC
```

AnnoDomini Graphical User Interface.

In COBOL programs, dates are represented using the data types and operations of the source language: numbers, strings of characters and flat records. Their intentional interpretation as representations of dates is not explicit. The AnnoDomini approach is based on reverse engineering the programmer-intended year interpretations, encapsulating them as abstract types, and replacing their implementation by safe, improved code with the same interface. This is done in three conceptual phases: seeding, type checking, and conversion.

In the first phase the user seeds chosen variable declarations of the program with year information; that is, some of the

the same type. During propagation AnnoDomini also checks that the seeded and propagated types are consistent with each other. For example, if the program assigns the contents of one variable to another, and the variables have different types, AnnoDomini signals an error.

In general, type errors may stem from several sources. For instance, seeding might be wrong, or the program may actually contain a Year 2000 Problem. AnnoDomini does not attempt to guess what the real cause of a type error is and how to eliminate it. It suggests a number of plausible corrective actions, however, typically to change the declaration of a variable, or to insert a type coercion, eg a coercion between 2-digit years and 4-digit years. Like type declarations, such coercions are specified as annotations – the actual conversion code is inserted automatically in the conversion phase. AnnoDomini also provides point-and-click access to the statements causing type errors and to the declarations of the variables appearing in these statements for manual browsing and editing of the source code.

AnnoDomini issues warnings for all relational and arithmetic operations for which there is insufficient type information to determine whether their operands contain years or not. This is a case where seeding is incomplete, with potentially dangerous consequences. The user is expected to check the warnings to determine whether they cover over any potential Year 2000 problems.

The third and final phase consists of virtual conversion and actual conversion. During virtual conversion the user specifies Year-2000-safe types for each variable. For example the user might specify that some variable should be expanded from a six-digit to an eight-digit date representation. Actual conversion is then fully automatic: at the push of a button, data declarations are expanded as desired, calls to the specified coercions are inserted, and arithmetic and relational operations involving two-digit years are replaced by calls to Year-2000-safe library routines.

AnnoDomini consists of three components: the analysis and conversion engine (60,000 lines of Standard ML), the graphical user interface (10,000 lines of Visual Basic), and IBM's Live Parsing Editor (a syntax-sensitive program editor). The three components are tightly integrated; for instance, by a single click the user can have code implementing a suggested corrective action inserted into the program automatically by the editor.

The underlying software reengineering method in AnnoDomini consists of identifying and isolating potentially problematic data and their associated operations according to their intended use (here as calendar years), encapsulating them as abstract types, and finally replacing their implementation by safe, improved code with the same interface. This method appears to be eminently applicable to other problems than Year 2000 remediation, such as reengineering financial systems for the introduction of the Euro or for the Dow Jones Index passing the 10,000 mark. To address such problems is one possible line of future work.

■

Please contact:

Morten Heine Sørensen – DIKU and Hafnium ApS
Tel: +45 35321405
E-mail: rambo@diku.dk

Software Renovation

by Arie van Deursen

In 1976, Belady and Lehman formulated their 'Laws of Program Evolution Dynamics'. First, a software system that is used will undergo continuous modification. Second, the unstructuredness (entropy) of a system increases with time, unless specific work is done to improve the system's structure. This activity of improving legacy software systems is called system renovation. It aims at making existing systems more comprehensible, extensible, robust and reusable.

Due to the fact that a typical industrial or governmental organization has millions of lines of legacy code in continuous maintenance, well-applied software renovation can lead to significant information technology budget savings. For that reason, in 1996 Dutch bank ABN AMRO and Dutch software house Roccade commissioned a renovation research project. The research was carried out by CWI, the University of Amsterdam, and ID Research. The goals of the project included the development of a generic renovation architecture, as well as application of this architecture to actual renovation problems.

Of the various facets of software renovation – such as visualization, database analysis, domain knowledge, and so on – an enabling factor is the analysis and transformation of legacy sources. Since such source code analysis has much in common with compilation (in which sources are analyzed with the purpose of translating them into assembly code), many results from the area of programming language technology could be reused. Of great significance for software renovation are, for example, lexical source code analysis, parsing, dataflow analysis, type inference, etc.

Program Transformations

Software renovation at the source code level includes automated program transformations for the purpose of step-by-step code improvement. In this project, we successfully applied transformations to COBOL programs, dealing with goto elimination, dialect migration (between COBOL-85 and COBOL-74) and modifications in the conventions for calling library utilities.

To make this possible, we developed a COBOL grammar, instantiated the ASF+SDF Meta-Environment with this grammar to obtain a COBOL parser and pretty printer, and designed term rewriting rules describing the desired transformations. The resulting system is capable of automatically performing the desired transformations on hundreds of thousands of lines of code, yielding a fully automatic transformation factory.

Object Identification

At a higher level of abstraction, software renovation includes the migration of legacy code to architectures better capable of meeting today's requirements. A typical example is the migration of procedural COBOL code to object technology. This is a process which cannot be fully automated. Instead, renovation tools will have to focus on helping the human reengineer in understanding the legacy system.

Thus, such renovation tools will have to extract as much meaningful information as possible from a legacy system, showing it to the human reengineer in a concise and usable manner. For the purpose of object identification, this information consists of the business data items (candidate object attributes) the programs or procedures performing the key tasks (candidate methods) and an overview of the combined use of these (candidate classes). We were able to develop heuristic techniques based on cluster and concept analysis to extract such class proposals automatically.

Conclusion

The overall result of the project is a generic renovation architecture aimed at program transformation and system understanding. A specific instantiation for COBOL has been developed, which has been applied to various real life case studies.

Pointers to publications with more information can be found at: http://www.cwi.nl/~arie/resolver/

■

Please contact:

Arie van Deursen – CWI
Tel: +31 20 592 4075
E-mail: arie@cwi.nl

Symbolic Techniques for Program Analysis

by Henk Nieland

As a consequence of our ever increasing dependence on the proper functioning of software systems, the need for proving their reliability has to be taken serious indeed. Until recently such proofs were only feasible for systems whose size is far below what is found in practice. Recently developed techniques, however, may offer a solution. The aim is now to apply these techniques effectively, so that real-life systems can be successfully analyzed. By combining mathematical rigour with manual computation CWI developed methods which can prove 'beyond reasonable doubt' the reliability of software systems of a realistic size.

In principle there are three approaches: proof by manual labour, fully automated proof techniques, and a mix between the two. CWI has shown that the last way enables the analysis of realistic systems.

Manual proof can be carried out in the context of Process Algebra (PA). The use of PA has advantages above other verification methods such as modal or temporal logic because of its high level of abstraction and its composition properties. A basic tool is the Cones & Foci Theorem, which effectively can be used to prove statements of the form: Specification = Implementation. On the basis of PA, CWI developed μCRL (micro Common Representation Language). The idea was to create a basis for sharpening symbolic techniques, rather than adding another language to the repertoire. With μCRL one can carry out proofs manually following strict logical rules. In practice, however, several such proofs remain 'sloppy', because the manual method is effective only for small systems, not exceeding one page of code.

Fully automated proof techniques are usually based on state automata. At CWI now systems with 10^8 states can be dealt with (in general the limit is 10^6), but realistic systems are still considerably larger. An even not really large system such as the software used for the safety of a small railway-yard, which was recently analyzed by CWI using propositional logic, consists of about 10^{1000} states. Of course, it is of the utmost importance to find ways to reduce the number of states. Our research indicates that by transforming processes described in μCRL to a normal form (Linear Process Operation) using rewriting techniques (automated induction, tree automata), exponential reduction of the number of states can be reached.

Since neither purely manual, nor purely automated techniques can cope with realistic systems, a compromise must be sought. By using proof checkers, which guarantees the required precision, in combination with manual control, CWI has reached promising results. Experience with checkers like Coq, PVS, and Isabelle, used in this way, shows that this approach can be effective for middle-sized systems. One may compare this hybrid technique with the way packages such as MATLAB and MAPLE are used in mathematical formula manipulation. This approach to putting formal proof techniques to practical use may in due course very well lead to a revolution in mathematical argumentation, as was foreseen already some thirty years ago by the eminent Dutch mathematician N.G. de Bruijn when he created his Automath system.

Meanwhile several instances of faulty software have been revealed by applying formal proof checkers under manual control. Recent Dutch examples include the automated control system for the legs of car lifting installations in garages, and for the doors of the dam in the Nieuwe Waterweg which protects the Rotterdam area by closing the doors in case of flood. Sincs many more such instances can be expected to show up in the near future, we may see before long the birth of a new profession: that of software prover.

More information can be found at http://www.cwi.nl/~jfg/

■

Please contact:

Jan Friso Groote – CWI
Tel: +31 20 592 4232
E-mail: JanFriso.Groote@cwi.nl

Security Verification: a Programming Language Approach

by Thomas Jensen

Electronic commerce with its use of programmable smart cards and payment via Internet must guarantee the confidentiality and integrity of the data involved in the transactions. The ever-increasing presence of software in these applications means that verifying that this software conforms to such security requirements becomes an all-important task which is far from trivial. The Lande research team at IRISA (Inria-Rennes) studies formal techniques for verifying security properties of applications written in the Java programming language and its dialect Java Card, destined for smart card programming.

A number of programming languages incorporate facilities for rendering a program secure eg, by protecting data from unwanted access or by limiting the capabilities of parts of code whose behaviour cannot be trusted. Using a high-level language to express the security management in a program (as opposed to relying on low-level or hardware mechanisms) facilitates formal reasoning about its correctness and opens up the possibility of using well-established techniques from programming language semantics to structure this reasoning. A recent example is Java that comes equipped with a complex security architecture which includes visibility modifiers to limit the accessibility of members of classes, the use of class loaders to create separate name spaces, granting of user-defined permissions such as reading and writing files, and dynamic checks that the executing code has a given permission.

The aim of our research is to develop methods that allow to verify security claims of such applications in a formal manner. This involves two activities: the formalisation of what a security claim is and a semantic model of the Java security architecture against which these security claims can be checked. Our initial effort has focussed on control-flow-based security that for a given code traces back in the execution history to discover on whose behalf it is executing, in order to check that those who originated the current operation indeed have the right to do so. Prior to verification, a program is submitted to a type analysis that for each (virtual) method invocation in the program returns an approximation of the set of concrete methods to which this invocation can correspond – this results in an approximate control flow graph for the program. From this graph we derive a transition system where the states are call stacks and where transitions are method invocations. The security properties to verify are formalised using a temporal logic that describes the allowed paths that can be taken in this transition system. The transition system is infinite and we rely on a novel reduction technique that for a given property allows to restrict attention to a finite part of the transition system in order to verify the validity of the property. The actual verification then becomes a classical model-checking problem.

Ongoing research in the group aims at extending these results in two directions. First, the current method requires that all of the program to verify must be present. This is a limitation in a world where code is loaded dynamically over networks. In order to solve this problem we are looking at how to modularise the various static analyses involved. The aim is a technique that for each unknown piece of code derives a security interface, ie, a security property that an imported piece of code must satisfy in order to be loaded. Second, we are in the process of applying this technique to the Java Card language for programming smart cards. Java Card is derived from Java by removing a number of language features that are too costly and not strictly necessary to implement on the resource-limited smart cards (no multi-threading, no floating-point values, no dynamic class loading etc.). The security model is somewhat different in that applications are completely isolated and communicate via explicitly shared objects. This activity is conducted in collaboration with Bull via the GIE Dyade between INRIA and Bull, and in the INRIA-sponsored research action Java Card, co-ordinated by the Lande research team.

More information on the Lande team can be found at http://www.irisa.fr/lande/. ■

Please contact:

Thomas Jensen – IRISA
Tel: +33 2 99 84 74 78
E-mail: Thomas.Jensen@irisa.fr

Deductive Proof of Software Properties

by Patrizia Asirelli and Franco Mazzanti

The aim of an recently begun IEI project is to experiment with the idea that a deductive approach can successfully be adopted to support the verification of properties of programs written in high level languages.

The use of high level languages (or better, the use of safe subsets of them) is being increasingly recommended by regulating organisations and standards for the development of critical software. However, when we try to nail down a precise definition of which properties should be automatically verified during the development process, we find a wide set of alternative definitions of safe subsets, different degrees of supported automatic verifications, and few available tools that might help the programmer (or verifier) in his/her task.

Because of the intrinsic fluidity of the problem (the same project might be constituted by different fragments with different criticality levels and for which different sets of properties should be guaranteed), we cannot claim that this kind of verification should be performed by the compiler, even if it actually needs almost all the information usually available to the compiler. The alternative, with which we are now experimenting, is to use a deductive environment, able to make use of all the information the compiler can gather on the program under analysis, for expressing and verifying the set of properties in which we are interested.

The high level language being considered in this project is Ada. This choice has automatically build a logical database containing all the basic properties of the program itself, as provided by the Ada compiler through its ASIS interface (the program syntactic structure, the static semantic relations between its components, and whatever else might be interesting to extract). This basic database is enriched with rules expressing more complex properties, usable by the verifier to flexibly compose logical queries about the program under analysis.

```
procedure MAIN is
   L: Integer;
   M: Integer;
   N: Integer;
begin
  if L = 0 then
    M := 10;
  end if;
  L := M;
  while L > 0 loop
    L := L - 1;
  end loop;
end ;
```

```
...
declaration(main_m_10_1,a_procedure_body_declaration ).
first_body_declarative(main_m_10_1, main_12_1).
next_body_declarative(main_12_1, main_27_1).
next_body_declarative(main_27_1, main_42_1).
next_body_declarative(main_42_1, nill).
last_body_declarative(main_m_10_1, main_42_1).
first_statm(main_m_10_1, main_62_1).
next_statm(main_62_1, main_103_1).
next_statm(main_103_1, main_123_1).
next_statm(main_123_1, nill).
last_statm(main_m_10_1, main_123_1).
...
```

A small Ada program and a fragment of the corresponding logical database.

been made for several reasons. Ada subsets are widely used for the development of critical systems (especially in the avionic/space field). Ada is very suitable for developing further advanced static verifications because of the richness and intrinsic safety of its type system. Ada comes with a very important draft ISO standard (ASIS) which defines a standard interface allowing a compiler to export all its knowledge on a program towards other development tools. Finally, good quality free development environments for Ada exist and are widely used.

The deductive environment used by the project is based on Gedblog, a deductive database management system developed at IEI and actively maintained. The Ada compiler used in the project is the GNU based GNAT compiler, and the ASIS library is a prototype version developed at EPFL (Ecole polytechnique fédérale de Lausanne).

The approach with which we intend to experiment in the project is the following. First, given an Ada program we plan to

The project activity has just begun and is currently in a preliminary study and experimentation phase. If the results are encouraging, we plan to continue the activity with the verification of more complex properties, like the absence of particular classes of run-time errors, or the full adherence to some particular 'safe coding guidelines' (whose satisfaction is too often still verified in a non mechanical way). We are open and looking forward to possible collaborations with other ERCIM partners interested in similar aspects. ■

Please contact:

Franco Mazzanti – IEI-CNR
Tel: +39 50 593 447
E-mail: {asirelli,mazzanti}@iei.pi.cnr.it

Certification of Imperative Programs in the System Coq

by Jean-Christophe Filliâtre

The system Coq is a proof assistant developed by the Coq team at INRIA, so far used to formalize mathematics and to prove the correctness of purely functional programs. From now on, it may also be used to establish the correctness and the termination of imperative programs (in fragments of C, Pascal, or ML).

The Coq Proof Assistant (see ERCIM News number 32) is a tool for specification and formal proofs, based on a highly expressive logic, the Calculus of Inductive Constructions. Therefore, it is naturally suited for mathematical formalizations and proofs of purely functional programs, since those are already terms of the logic. But the certification of programs is not realistic without dealing also with imperative programs, so a module of certification of imperative programs has been recently introduced in the system Coq.

The programs are given in an ML-Pascal dialect mixing imperative features (references, arrays, while loops, sequences) and functional features (functions as first-order objects, polymorphism, recursive datatypes). They are specified in a Floyd-Hoare logic style, by insertion of logical assertions, such as pre- or post-conditions or loop invariants. Termination is justified by the insertion of a pair variant/relation associated to each loop or recursive function. Then an automatic tactic takes a specified program and produces some proof obligations, whose validity implies both correctness and termination of the initial program.

The method involved is based on a functional translation of imperative programs. Starting with an annotated program, we first determine its effects (access or modification of references or arrays). Then, using this information, we build a proof of its specification, whose skeleton is a functional translation of the imperative program which expresses its semantics. This proof is of course incomplete, and each 'hole' will correspond to a proof obligation. Indeed, this partial proof term is given to a specific tactic which proves the specification by generating a proof obligation for each missing part of the proof term, in a way similar to the type-checking conditions (TCC) of the PVS system. It is important to notice that this functional translation, and the corresponding proof term, are completely hidden. The user only sees the specified program he gave, and the resulting proof obligations. Then he can use all the power of the proof assistant to prove them. Once the proofs are done, the programs can be pretty-printed in C or ML code to be compiled and linked in bigger applications.

This technology is already distributed with the system Coq, and has been applied on quite complex algorithms (select, quick sorting algorithms, Knuth-Morris-Pratt string searching, ...). The interests of such an approach are mainly the use of a highly expressive logic and the use of a secure proof assistant with a great experience in formal proofs.

There is still some work to be done to reach a fully operational certification environment for imperative programs. A first improvement will be an extension of the programming language, with an addition of exceptions and other imperative datatypes such as records for instance. Another improvement, which is essential, concerns modularity, since there is no big software development without a good notion of modules. So we have to understand what is a good notion of module with respect to specified programs, and in particular what are the visibility rules associated to those modules.

For more information about the Coq project and the certification of imperative programs in the system Coq, see web site: http://coq.inria.fr/

■

Please contact:

Jean-Christophe Filliâtre – INRIA
Tel: +33 1 69 15 64 53
E-mail: Jean-Christophe.Filliatre@inria.fr

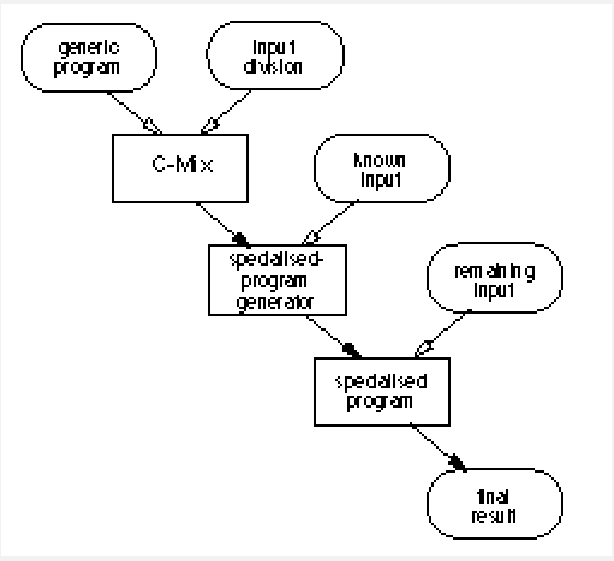
Generating Program Generators

by Arne J. Glenstrup, Henning Makholm and Jens Peter Secher

Research in semantics based program manipulation has been carried out in Copenhagen for more than a decade. The DART project (a member of ERCIM partner DANIT) is putting theory into practice by deriving program manipulation tools from programming language theory. An important instance is that program generators (and generator generators, etc.), can be produced automatically and correctly from executable problem specifications.

One of the main goals is to produce tools that are automatic in the sense that once instantiated by a user, they can be repeatedly applied to widely varying problem instances without further user intervention. This semantic approach is applicable to a wide range of areas:

- program transformation can be safely performed, eg for instrumentation, simplification, or translation
- optimisation by specialisation of generic programs to specific tasks can significantly reduce computational overhead
- generation of compilers from interpreters can be achieved automatically.



Program specialisation with C-Mix.

Partial Evaluation

Partial evaluation is an automated source-to-source transformation technique that can be used to optimise generic software: a program together with some known portion of its input, is transformed into a specialised version obtained by pre-computing the parts of the program that only depend on this input.

Usually when writing programs there is a tradeoff between, on the one hand, generic and easily maintainable programs and, on the other hand, fast programs. Partial evaluation bridges this gap, letting you have your cake and eat it too: generic programs are (automatically!) turned into fast programs for specific problem instances.

In a software development setting, a generic, well-tested module can be taken off the shelf and then specialised automatically to a specific usage pattern, thereby speeding it up. Since the specialisation preserves the semantics of the module, it is possible to produce both reliable and efficient software.

Specialisation can be done quickly, and this means that rapid prototyping and production can be unified. As a side-effect, some of the optimisation usually done by hand can be done in an automatic, application-independent way. Furthermore, the original program can be left untouched, so its structure and readability can be retained.

Theoretical foundation: the Futamura Projections describe the generation of compilers from interpreters, avoiding the error-prone task of writing a compiler by hand. DART researchers were the first in the world to transform these theoretical ideas into practice by producing an automatic compiler generator using these principles. Compiler generation has continued to be a focus point of DART work in partial evaluation.

Partial Evaluators

The most sophisticated partial evaluators today work with semantically well-defined languages like Scheme and SML (an example is Similix; see web address below). The languages that are more widely used in the software industry tend to be less rigorously defined, but nevertheless the understanding gained by semantics-based techniques have proved valuable in the development of practically oriented partial evaluators like DART's C-Mix for the C language; FSPEC for Fortran; and a partial evaluator for Java that is under way in the project.

There is an interaction and exchange of ideas with INRIA. In Rennes, the COMPOSE group develops the Tempo Specialiser for C which complements C-Mix by focusing on a very fast specialisation process that makes it possible to produce specialised object code at run time.

Successful experiments have been carried out in a number of settings, including ray tracing, model simulations, implementations of domain-specific languages, numerical algorithms, and compiling by specialising interpreters.

C-Mix: Partial Evaluation of C

C-Mix is an automatic partial evaluator that specialises C programs that conform to the ISO C standard. It takes a generic program together with a classification of the input and produces a specialised-program generator. When this generator is provided the portion of the input that is known in advance, it will produce a specialised program, as depicted in the figure. When the specialised program is run on the remaining input it produces the same output as the original program, only faster. Often, the same specialised program is reused, resulting in significant speedups in the running times.

The specialised-program generator is a stand-alone program that can be used without any knowledge about partial evaluation.

C-Mix does a number of complex analyses on the subject program – eg, alias, liveness and binding-time analysis – to calculate which constructs depend on known input only. Using this information, the specialised-program generator can be produced.

C-Mix is available free of charge, and is implemented with standard tools (GNU C/C++ compiler and tools), which means that it is highly portable (it runs on Unix-like systems, Windows, DOS, etc.). It has an HTML-based inspection toolkit that provides the developer with the results of the analyses, enabling fine-tuning of the specialisation process. URL http://www.diku.dk/topps/Research.html contains links to C-mix, Similix, and various other DART activities.

The COMPOSE group in Rennes also work on program specialisation and provide a C program specialiser, see http://www.irisa.fr/compose/. ■

Please contact:

Jens Peter Secher – DART
Tel: +45 35 32 14 08
E-mail: jpsecher@diku.dk

PROGGEN – a Tool for Automatic Code Generation

by Therese Nilsen

ProgGen is a generic tool for automatic code generation from any textual design language. ProgGen was developed by SINTEF, and the first version came in 1992.

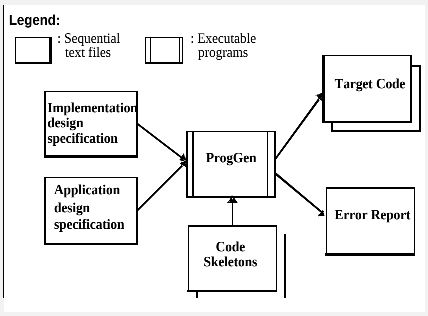
ProgGen has until recently only supported SDL'92 and SDL'88 (more exactly the textual representation and not the graphical representation), but has been extended to any language that can be described using the BNF notation and is of type LALR-1. ProgGen is a flexible transformation tool which can be used to produce customized code generators.

The code skeletons implement user-defined strategies (or rules) for transforming SDL or other design language descriptions. By 'strategy,' we mean that it is possible to implement a design-language specific concept in many different ways. Program code, but also other kinds of output (makefiles, test suites, documentation) can be generated. You do not need to develop new code skeletons for each new application; it is not necessary if the implementation strategies remain unchanged, or if the generated code is platform independent (generic). We can provide customers with help for developing skeletons.

ProgGen has been used for some years by several industrial users in Norway (Stentofon, Alcatel) and elsewhere (Alcatel Bell). Existing applications of ProgGen include a variety of code generators for the programming languages C, C++, CHILL and ADA. The tool is fully flexible and other target languages (eg Java) are also possible.

ProgGen is implemented in C, and will run on any platform with a C compiler, support for the portable C library, and adequate main memory for storage of the internal representation of the SDL model. We deliver ProgGen for UNIX based workstations. However, existing users have also installed it on PCs and VAX/VMS.

These days we are working with a project to extend ProgGen to allow for information about implementation design and let this information control the transformation. We wish to demonstrate



ProgGen and its environment.

that implementation design can be separated from application design. The advantage with this approach is the possibility to use one implementation design specification on different application specifications. Likewise, different implementation design specifications can be used with one application specification.

More information on ProgGen at: http://www.informatics.sintef.no/~proggen/main.html

■

Please contact:

Therese Nilsen – SINTEF
Tel: +47 73 55 0359
E-mail:
therese.nilsen@informatics.sintef.no

MAP: a Tool for Program Derivation based on Transformation Rules and Strategies

by Alberto Pettorossi, Maurizio Proietti, and Sophie Renault

Since 1987 the Department of Informatics of the University of Rome Tor Vergata and the IASI Institute of the National Research Council (CNR), Rome, have been cooperating on the development of techniques and tools for automatic program derivation and validation. This work has used a transformation methodology based on the so-called 'rules + strategies' approach.

The basic idea for this approach goes back to the seminal papers by Burstall-Darlington in 1977 (for the case of functional programs) and Tamaki-Sato in 1984 (for the case of logic programs). These papers show how a given specification, written as a set of recursive equations or a set of Horn clauses, can be transformed into an efficient program by applying suitable transformation rules which are guaranteed to preserve the intended semantics. The application of these rules should be guided by suitable strategies that, for some classes of initial specifications, allow us to derive efficient programs.

We are currently developing a tool, called MAP, to support the interactive derivation of logic programs by means of transformation rules and strategies. The MAP system has been implemented in SICStus Prolog and its graphical user interface has been developed using Tcl/Tk.

At present the MAP system provides a menu with a set of predefined transformation rules which include: definition introduction, definition elimination, unfolding, folding, goal replacement, generalization, and case split. If suitable conditions are satisfied, these rules preserve the least Herbrand model semantics.

In MAP the programmer also has a menu with a set of predefined strategies, ie, sequences of applications of transformation rules. Strategies are needed to derive programs with specific syntactic properties, such as: tail recursion, linear recursion, absence of existential variables and unnecessary data structures and absence of redundant nondeterminism. These syntactic properties make the derived programs very efficient in time and space.

The program derivation process requires some theorem proving capabilities. In particular, in order to apply the goal replacement rule we need to show the equivalence between a new goal and one to be replaced. For instance, during program derivation we may need to use the associativity of list concatenation, which is expressed by the goal equivalence:

$$\exists C\ (app(A,B,C) \wedge app(C,D,E)) \leftrightarrow \exists F\ (app(B,D,F) \wedge app(A,F,E)).$$

Formulas of this type can be proved off-line and all equivalences can be stored in theories which represent our knowledge about the predicates used. The MAP system allows the programmer to create, load, update, and store theories which may be useful for the derivations at hand.

MAP keeps track of the history of the program derivations, and provides the user with some facilities for backtracking to previous programs and exploring alternative derivations. In MAP, program derivations can be operated on by, for instance, loading, editing, printing and saving them, and programs, theories, and histories can be restored.

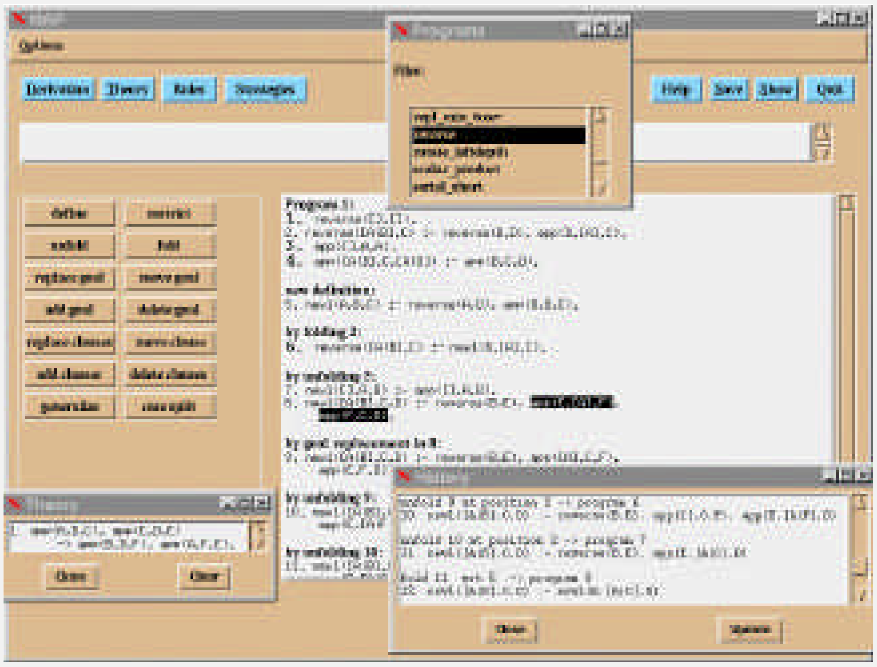
We are planning several enhancements to the current system. We would like to improve the ease of interaction with the user by providing more powerful graphical tools for navigating through the tree of alternative program derivations. We would also like to be able to extract program derivations and to reuse them for deriving in a (semi-) automatic way new programs from similar initial specifications.

We are currently working to extend the system to other languages, and in particular, to: general logic programs with negation, constraint logic programs, and functional programs. We are developing libraries which allow the user to load several sets of predefined rules, strategies, and theories into the system. We also plan to design languages that enable the users to define their own rules and strategies, so that the system may work as a generic, programmable program transformer.

In addition, we intend to include modules to support automated theorem proving and program analysis. The future MAP system should be able to exploit the information produced by these modules for performing very powerful program transformations whose applicability conditions may depend on the specific properties of the programs at hand.

As already indicated in the literature, there are various applications of program transformation within the field of machine-supported software production, reuse, and validation for which the MAP system could be a useful supporting tool. Among these applications, we should like to mention program specialization, program synthesis, and program verification.

Our research is currently supported by the Italian Ministry for the Universities and Scientific and Technological Research and by CNR. More information is available at:
http://www.iasi.rm.cnr.it/~{adp,proietti} ■



Interactive program derivation using MAP.

Please contact:

Alberto Pettorossi – University of Rome - Tor Vergata,
Maurizio Proietti – IASI-CNR
Tel: +39 06 7716426
E-mail: {adp,proietti}@iasi.rm.cnr.it

Sophie Renault – Université de Montréal
E-mail: renault@IRO.UMontreal.ca

Vanilla: Towards more Modular Programming Languages

by Simon Dobson

The emergence of the Internet as a commercial force is now well under way. A particularly powerful business model is the notion of a 'virtual enterprise', a temporary alliance of companies brought together to exploit some transient market opportunity. Such systems typify a class of applications in which a set of components must be composed in a highly dynamic manner. As well as the obvious correctness issues, it is clear that notions such as webs of trust, role-based interactions and on-the-fly adaptability are neither properly understood nor well supported by current programming language technologies.

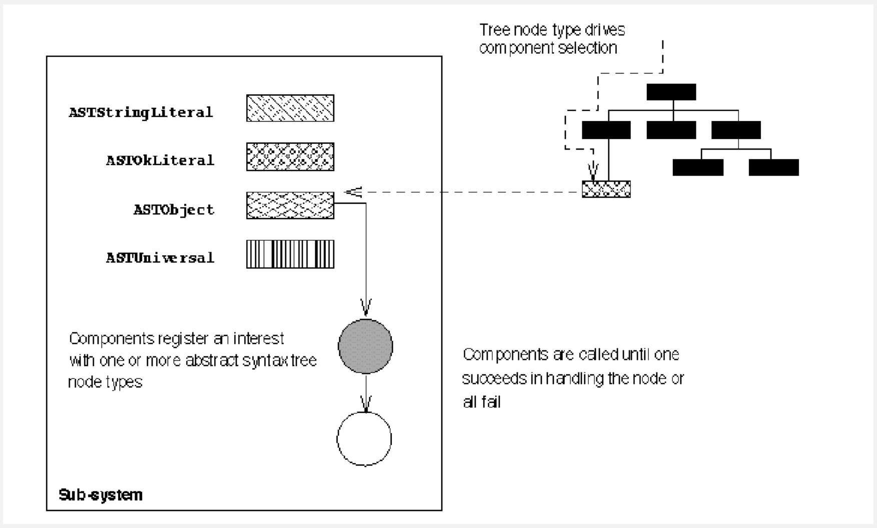
Trinity College Dublin has a long-term interest in very large-scale distributed systems. We felt that there was a need to experiment rapidly with novel language constructs for component composition, object re-use models and non-standard type checking. We were also concerned by the change in programmer demographics away from dedicated programmers happy with mainstream languages and towards domain specialists needing targeted, high-level scripting languages in which to write their own applications. Traditionally, programming language research has had an unacceptably high barrier to entry. A tool such as compiler or interpreter is a large, complex software system, which is difficult to understand and modify. This makes it difficult to experiment with minor changes to full languages, or to create 'bespoke' domain-specific languages.

The Vanilla project applies component-based program composition to the construction of the language tools themselves. Vanilla allows the language designer to construct a language by combining a number of components (or 'pods'), each of which captures the syntax, type-checking and behaviour of a single language feature. A typical pod might define a feature such as the integers with a particular concrete syntax for the numbers and operators, a parser to the corresponding abstract syntax, a type checking component to ensure the type-correctness of programs and an interpreter component to evaluate the expressions. The individual components may be changed independently, for example allowing a new concrete syntax to be defined while re-using the existing types and behaviours. New features may be defined in isolation before being combined with other features to create the final language. This incremental, experimental approach greatly reduces both complexity and development times.

A common design pattern used within Vanilla (shown in the figure) allows components to express an interest in one or more abstract syntax tree node types. On encountering such a node the system walks the chain of interested components, allowing each in turn to either deal with the node, determine an error condition, or 'pass' the node to the next component in the chain.

We have constructed a number of standard pods covering a large portion of the language design space – including procedural, functional, object-oriented and 'typeful' language features. It turns out that many features are orthogonal to each other, so (for example) it is possible to define object types without committing to exactly what types may compose the members. This makes for some interesting generalisations to well-known concepts.

Of particular interest are the pods handling network interactions, for example with CORBA objects. We use a component-based mapping between Vanilla and CORBA IDL, which allows applications to access objects on the Internet with the absolute minimum of additional code. Moreover, by completely separating the protocol handlers from the rest of the application, we allow the same code to be used simultaneously via a number of different protocols (CORBA, DCOM, event-based models etc). This radically simplifies the construction of applications in highly heterogeneous environments.


Tree node type drives component selection
ASTStringLiteral
ASTOkLiteral
ASTObject
ASTUniversal
Components register an interest with one or more abstract syntax tree node types
Components are called until one succeeds in handling the node or all fail
Sub-system

Common design pattern used within Vanilla.

We are currently using Vanilla to explore four complementary language areas: mobile objects and agents, novel object composition patterns, dynamic construction of applications based around XML document, and scripting languages for describing interactions with intelligent buildings. More information on the Vanilla project at http://www.cs.tcd.ie/Virtues/Vanilla/ ■

Please contact:

Simon Dobson – CLRC and Trinity College Dublin
Tel: +353 1 608 2224
E-mail: simon.dobson@cs.tcd.ie

Dynamic Translator Development: Modelica in the Python TRAP

by Thilo Ernst

Modelica is the new unified, object-oriented description language for dynamical models of physical systems developed in an international effort in which GMD co-operates. GMD is developing a Modelica translator for integration in the Smile dynamic simulation environment. The Python language which is used as an integration platform in Smile, together with associated tool components also proved to be a powerful basis for translator development. By its combination of very high level of abstraction, interpreted execution, and ease of extensibility, Python enables a new development methodology also for language processors; with reference to generic simulation environments, it provides a unique foundation for R&D in dynamic model evolution.

Modelica (http://www.modelica.org) is a unified language for dynamic models of complex physical systems being developed in international effort (formally, a combined EUROSIM technical Committee/SCS Technical Chapter) in which GMD participates (see also ERCIM News Number 32). Smile (http://www.first.gmd.de/smile) is an object-oriented dynamic simulation environment developed by Technische Universität Berlin and GMD. In its latest revision, it heavily builds on Python as an integration platform for both external and internal software components. GMD is developing a prototypical Modelica compiler component for integration into Smile. The Modelica processor is intended to be open, extendible, and reusable.

Python (http://www.python.org) is an interpreted, object- oriented language often referred to as a 'scripting', 'extension' or 'glue language', as it is well-suited and popular as a framework for integrating software components across diverse implementation languages and programming paradigms. However beyond that, Python is a full-fledged, platform-independent, modern programming language. Python offers powerful features such as classes, modules, exceptions, dynamic typing and very high level collection data types in a concise, regular and very readable syntax. The language's high level of abstraction and (byte-code-)interpreted execution provide an outstanding development efficiency. Performance bottlenecks can easily be identified and can be effectively attacked by 'extensions', ie optimised low-level (C/C++) re-implementations of the (typically small) code parts in question. This enables a very efficient rapid prototyping/rapid application development (RP/RAD) methodology. The Python language and software package are free and not subject to legal restrictions hampering any kind of application. Python is mature and reliable, and (thanks to a large and active user community) a huge collection of library components written in or interfaced with Python exists, for the most part free like Python itself.

Using Python for translator implementation was a rather obvious idea in the project setting described above, as a Python interface between the Modelica translator and the Smile-internal equation system data structure was planned anyway. Therefore this option was evaluated in more detail. It turned out that the language has a set of features that very effectively can be exploited to approach common data structure and algorithm patterns found in compilers. For instance:

- Python offers sequence types (lists and tuples) and their standard manipulation methods as built-ins. List manipulation is sufficient to implement algorithms based on incremental set manipulations, which are ubiquitous in compilers.
- Python's object model and extension concept make it easy to work with sets or graphs of objects (that represent any semantically relevant information) using a class library, and later on transparently migrate to a more efficient bitset implementation.
- Python's dictionary datatype can be used to represent arbitrary mappings between Python objects. Mappings occur frequently in compilers as well: *name space* and *symbol table* both refer to mappings (on different levels of abstraction).
- Python's object model can be effectively used for object oriented compiler techniques, eg representation of abstract syntax tree (AST) node sorts by a class hierarchy in which standard functionality (eg tree traversal according to the *visitor* design pattern) is packaged. Python is not statically typed, but has a dynamic type system which can be easily used to enforce compiler-specific constraints, eg local wellformedness constraints for AST nodes can be checked in the node constructors.
- Python, by integrating concepts from both worlds in one language, also allows the user to choose the right mixture of functional and imperative programming adapted to the task at hand.
- Python uses a reference-counting based automatic memory management scheme hidden from the user: Objects can be simply created without caring about the memory allocation that is automatically happening; objects are silently reclaimed as soon as they are no longer referenced.

The transformation phase of a translator is where these advantages can be best exploited; lexical and syntactical analysis of the source text have to be done already. Fortunately, front-end tool components (scanner and parser generators) already were available as Python modules, so only a thin layer of tooling needed to be added to obtain a small, but sufficiently powerful Python-based development environment called TRAP (Translator RApid Prototyping) for building a Modelica translator. TRAP takes a compiler description consisting of an EBNF-style grammar specification (enriched with semantics actions expressed by pieces of Python code

```
compiler SimpleMod

   # comment syntax of language processed
   comment   r'//.*$'
   comment   r'/\*.*\*/'

   # lexical tokens
   tokx`zNT '[A-Za-z][A-Za-z0-9_]*' # default semantics: matched text

   token INT_CONST '[0-9]+':
     string.atoi(str)     # explicit semantics: convert to IntType

   # grammar: nonterminals with rules & semantics
   nterm primary
      # default: pass through constituent's semantics
      <- INT_CONST
      <- "time"
      <- "false"
      <- "true"
      <- component_reference
      <- "(" expression=E ")":
         E  # explicitly pass through semantics of E

   nterm name::[]  # type constraint: Python list
      <- ["."   IDENT+]  # non-optional repetition with separator "."
                       # automatic semantics: list of strings


   nterm class_definition::Mclass  # type constraint: a node class
      <- class_key=C IDENT=i1 Mcomment=K (component*)=T "end" IDENT=i2 ";":
         if i1 != i2:   # a simple semantics check
            ERR("Name mismatch:", i1, "/", i2)
         Mclass(C, i1, K, T) # construct AST node as semantics value


   # abstract syntax: a node type definition
   Mclass (
      key,   # default field type: StringType
      name,
      Mcomment,
      components::(component) # type: Python tuple of 'component' nodes
   )
```

Example constructs of the TRAP description.

attached to the production rules) and a concise hierarchical description of AST node sorts. In addition, type constraints for non-terminals and fields can be specified. From this, a (Python) *compiler frame module* is automatically generated, providing scanner, parser, the set of node class definitions with standard method instrumentation (printing, dynamic typecheck, traversal, pattern matching) and some auxiliary code. That way, source code of the language to be processed is easily converted into Python data structures; subsequent transformations are implemented directly in Python. The figure presents sample snippets from a compiler description.

Modelica can be categorised as a special-purpose language of medium complexity. TRAP of course should be generally useful for building translators for such languages. Indeed, it was already used for bootstrapping itself. However the emphasis here is not on building yet another compiler tool – TRAP mainly integrates relevant concepts from existing toolkits such as Cocktail, Gentle, and PCCTS. The important point is that this integration was done in the open, dynamic framework provided by Python.

In the context of processing Modelica, expressing semantics transformations in Python is more than merely convenient - this approach indeed provides a unique foundation for further R&D in generic modelling and simulation: Currently, most generic simulation systems have a rigid separation of compilation vs. Simulation phase. However, for certain application classes, it would be desirable to change structure and/or details of the model being worked on during simulation. With a Python-based, dynamic Modelica translator, (re-)doing semantically complex model transformations at simulation time poses no technical problems. A fully dynamic modelling and simulation system architecture in which concepts like dynamic model evolution can be conveniently investigated is an important area of future work. ■

Please contact:

Thilo Ernst – GMD
Tel: +49 30 6392 1919
E-mail: Thilo.Ernst@gmd.de

New Language on the Block: Java for High-Performance Computing?

by Mike Ashworth

The Java language is not only an emerging new technology in Web-based computing, but is starting to have a considerable influence in many IT-based science and engineering application areas. For example, it is emerging as a serious contender for use in High Performance Computing (HPC). At the CLRC Daresbury Laboratory, we are developing a multi-language approach in which Java is used as a front-end to existing HPC programming environments. This allows the utilization of high-performance legacy Fortran and C codes within a Java wrapper that facilitates the construction of Graphical User Interfaces (GUI) and access to Web-based client-server computing.

The Java language was designed by James Gosling at Sun Microsystems starting in 1990. It was first intended as a new language for embedded systems in consumer microelectronics, which led to the basic design features of Java: it is simple, object-oriented, architecture neutral, robust, secure and extensible. In 1993, as the World Wide Web was developing to a more graphically oriented interface, the Java team realised that Java also had ideal qualities for Web-based applications. Java has some similarities with C++, but it is much simpler. Many of the more advanced features of object-oriented programming, such as operator overloading, pointer arithmetic and multi-dimensional arrays, were left out.

How does Java work?

Java source is compiled to class files that contain machine-independent byte-code. This is like machine code in form but is not specific to any particular hardware. When a Java enabled client, such as Netscape or Internet Explorer, accesses a page containing a Java applet (a small Java application embedded in a Web page), the byte-code is downloaded and runs on the client's own hardware using an interpreter known as the Java Virtual Machine. Because Java is compiled to byte-code, it can run on any platform for which an interpreter has been written. Java is small so it can run efficiently on anything from PCs upwards, and the interpreter takes only a few hundred kilobytes.

Java and High Performance Computing

Java is more efficient than other scripts, but it is still very slow compared to traditional languages. Even if compiled to native machine code, there are features in the language that make it currently impossible for native Java compilers to apply the normal optimizations which allow Fortran and C programs fully to exploit the hardware of state-of-the-art. For example, security considerations in Java demand that all array references are checked for out-of-bounds indexes at run-time, which effectively kills performance for computational loop kernels. Experiments at IBM have shown that if this checking can be performed at compile-time and the compiler is allowed to make other optimizations, Java code improves from 1% of Fortran to 80-100%.

Vendors, software developers and computer users in the US have formed a group known as Java Grande, which believes that Java has the potential to be a better environment for HPC than any previous language. The Java Grande Forum aims to develop a consensus for the use of Java for Grande applications and to make recommendations for changes to the Java standard and for the establishment of standards for libraries and services. Interest is also growing in the UK. The First UK Workshop on Java for High Performance Network Computing was held at Euro-Par '98 in Southampton in September 1998. There are currently moves to form a UK Java Grande Forum similar in style to the US

Forum, but with a greater emphasis on the services and tools required by high performance applications.

Java, HPC and the Web

The World Wide Web was conceived as a means of accessing information, but it is increasingly evolving towards a means of providing services. A major sea change is happening on the Internet, due in no small part to the emergence of the Java programming language. Java is expanding rapidly and comes complete with a wide range of class libraries and toolkits for Web access, thread based computing and GUIs.

Although the popular image of the Web has been as a global information provider, this has not been its only manifestation. Increasingly, organisations are setting up an Intranet - a corporate Web server providing internal information and services. In an industrial R&D environment, the Intranet may enable access for everyone in a department to the technical designs and specifications of a new product together with the project specifications, modification records, test procedures, test results, marketing information etc.

It is in this environment that we see the major benefits of integrating Java, with its support for Web-based computing, with traditional high-performance programming using Fortran and C. This vision is of a new type of multi-language multi-paradigm programming environment for high performance applications which makes use of innovative Web-based and GUI technologies provided by Java and provides a high performance and simple API and a high level of portability and flexibility for large-scale applications. It emphasises the strengths of each component: Java allows easy access to Web-based computing and facilitates the construction of GUIs; Fortran and C allow large-scale applications best to exploit high performance hardware. ■

Please contact:

Mike Ashworth – CLRC
Tel: +44 1925 60 3663
E-mail: M.Ashworth@dl.ac.uk

A Formal Semantics and an Interactive Environment for Java

by Isabelle Attali

Being both object-oriented and concurrent, the Java model features interrelated aspects that are critical for the understanding of an application: objects, static variables, threads, locks, etc. We want to know how to visualize the various entities that participate in the execution of a Java program.

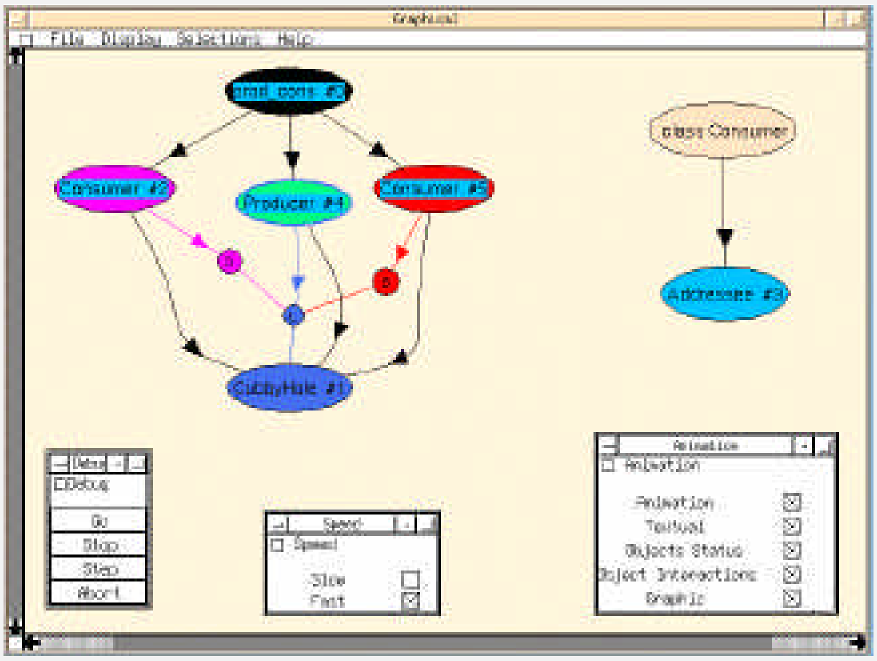
These concepts are not easy to understand and to handle for a programmer, nor easy to formally describe from a designer point of view.

We address these questions using a formal description of the syntax and the semantics of the Java language (useful for language designers and implementers), and as well as an interactive development environment (useful for programmers). We are using a powerful generic interactive environment named Centaur (http://www.inria.fr/croap/centaur/centaur.html). Centaur produces a programming environment, dedicated to a given language, from a formal description of the language.

This research started in spring 1997 with Denis Caromel (Professor at University Nice - INRIA Sophia Antipolis - CNRS) and Marjorie Russo, PhD Student), and is pursued at INRIA Sophia Antipolis.

The topology of instance variables within an object-oriented system can be rather complex. Are they mainly trees? Are there DAGs? Are there cycles, and where? Those are important questions for understanding an application. How can a programming environment provide that information to programmers in a simple and effective manner, without requiring them to compare reference values within a text-only debugger? Static variables raise special concerns, as they can be accessed from objects without an explicit reference to them.

When it comes to multi-threading and concurrent accesses, that information becomes even more crucial. How can we effectively visualize the objects shared by a given thread? Are there any recursive calls, either directly or through



View of the Java graphical environment.

cycles? Do several threads access a single object at the same time?

We provide solutions to those problems and demonstrate them in an interactive and graphical environment (shown in following Figure and also at http://www.inria.fr/croap/java). It allows us to visualize objects and thread activities at execution in both a textual and a graphical manner, eg, the topology of the object graph (trees, DAGs, cycles), thread activities and status, locks, and synchronizations. Synthetic views give relevant information on status of objects and interaction between objects and threads.

Using the Centaur system, this graphical environment is derived from an executable operational semantics of Java: execution a Java program is, in fact, an interpretation of the semantics with the program source as data (there is no byte-code, and no virtual machine is required). Animation of the various views is performed by messages from the semantics interpreter to the visualization engines (textual and graphical) in order to show, during execution, the changes in the structures. No instrumentation is required from the programmer.

We do not want to compete with commercial systems for Java development. Instead, we propose two main concepts which can then be exploited within any Java environment:

- graphical visualization of the object graph topology
- interactive symbolic debugging (no (re-)compilation is necessary).

Future activities include the following directions:

- allow user-control of the interleaving between all threads
- gain in scalability, with an abstract view of the object graph
- benefit from the formal definition the ability to perform formal verification of Java programs.

Related Projects

With Denis Caromel and Carine Courbis (PhD Student), we recently started a related research project. Our goal is to facilitate smart card programming using Java Card, a Java subset that deals with limited resources (http://www.inria.fr/croap/javacard).

Finally, with Denis Caromel and Romain Guider (PhD Student), we are also investigating the use of static analysis, based on abstract interpretation, for distributed and parallel programming in Java. For this work, we utilize the product of another related research project: the ProActive PDC Java Library (formerly Java//, Java Parallel), a Java library for seamless sequential, multithreaded and distributed programming using Java (http://www.inria.fr/sloop/javall). See also http://www.inria.fr/croap/java. ■

Please contact:

Isabelle Attali – INRIA
Tel: +33 4 92 38 79 10
E-mail: Isabelle.Attali@sophia.inria.fr

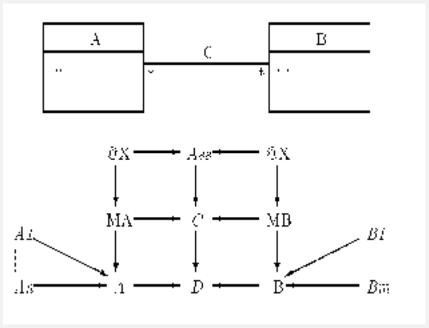
Formal Underpinnings of Object Technology

by Juan Bicarregui

Object-Oriented Analysis and Design notations, such as UML, are rapidly becoming the mainstream industrial approach to the development of software. They include a range of techniques for a wide spectrum of software engineering tasks from domain modelling to implementation. However, such techniques are generally without a formal interpretation which is essential for rigorous development as required by applications in critical areas, such as medical database and robotic systems, defence and chemical process control.

In collaboration with Imperial College, London, and Brighton University, we are developing a compositional formal interpretation of object model and statechart diagrams as used in OOA\&D notations. We interpret diagrams as logical theories in the Object Calculus (Fiadeiro and Maibaum 1991). Separate theories are constructed for separate diagram components such as object instances, class managers and associations which are then combined with categorical constructions to yield a modular definition of a whole system.

For example, the interpretation of an object class is constructed by taking the co-limit of theories for instances of the class and a general manager theory which handles the creation and deletion of instances. Associations are interpreted



Two classes related by an association and their interpretation. Theories for instances of the two classes, Ai and Bi, are combined with general class manager theories, MA and MB, to give the theories of the classes, A and B. A general theory for associations, Ass, is combined with the class manager theories via linking theories, @X, which relate the identifiers. The interpretation of the whole diagram, D, is the co-limit of the interpretations of the diagram parts.

by bringing together the theories of the associated classes with a general theory for associations via link theories that ‘glue’ the identifiers of the associated theories with that of a generic association. Annotations such as cardinality or aggregation constraints are interpreted as axioms in the co-limit theory. In this framework, aggregation and subtyping can be seen as special cases of the general concept of association.

The Object Calculus defines a notion of locality which ensures that only actions local to a particular theory can effect the value of the local attributes. Locality is a logical counterpart to object-oriented data encapsulation and the work has led us compare these two principles.

The formalisation has been used to justify some common transformations of class models such as those presented in the UML literature. By providing a semantically-based transformation calculus it is possible to obtain many of the benefits of formal methods without the need for users to reason directly in mathematical formalisms.

The incremental approach to the interpretation of diagrams makes the formalisation suitable as a basis for a support system which would provide a formal basis for the validation and verification of system specifications. ■

Please contact:

Juan Bicarregui – CLRC
Tel: +44 1235 44 6648
E-mail: jcb@inf.rl.ac.uk

The JoCaml System: a Language for Programming on the Internet

by Sylvain Conchon, Fabrice Le Fessant and Luc Maranget

The JoCaml system is an experimental extension of the Objective-Caml language with the distributed join-calculus programming model. This model includes high-level communication and synchronising channels, mobile agents, failure detection and automatic memory management. JoCaml enables programmers to rapidly develop distributed large-scale applications, obtaining both the Objective-Caml ease of programming and extended libraries, and the join-calculus distributed and concurrent features. It can already be viewed as the next-generation Internet programming language.

Distributed programming languages can be divided in two opposite kinds: concurrent calculi (CSP, CCS, pi-calculus, etc) and practical languages (JAVA RMI, DCOM, CORBA, Actors, etc). Concurrent calculi try to catch the formal basics behind distributed parallel programming, but are hardly implemented in really distributed platforms. Practical languages are heavily used, but lack formal specifications.

The PARA project at INRIA adopts both views. We thus designed a formal process calculus: the join-calculus and built practical implementations of programming languages based upon the join-calculus. We also made the significant effort of releasing tutorials and manuals.
Two implementations are currently available: the distributed join-calculus and the JoCaml system. In this note, we focus on the second, more efficient, and more promising implementation. Work on the first implementation is now frozen and it thus may serve as a stable platform for developing small distributed applications.

After a first development effort of six months, JoCaml has been made available on the World-Wide Web in May 1998 at a beta stage. New beta releases have been distributed more recently with enhanced distributed garbage collection, and a final release will be made in a few months when failure detection is terminated.

JoCaml’s programming model contains useful communication abstractions, such as high-level point-to-point channels, mobile agents and remote method invocations. Spaces and agents are uniformly organized in hierarchical trees, with failure detection and possible

```
Client --------------------------

let def view ps | init! () =
 let get_ps_viewer = Ns.lookup
        "ps_viewer" vartype in
 get_ps_viewer (top_location,ack);
 view(ps)
or view(ps) | ack!(viewf) =
  {viewf(ps);reply} | ack(viewf);;

{| init () };;

view "paper1.ps";
view "paper2.ps";;

Server  -------------------------

let def get_ps_viewer (site,ack) =
  let loc ps_viewer
             [PostscriptViewer] do
  { go(site);
    ack(PostscriptViewer.view) }
  in reply;;

Ns.register "ps_viewer" ps_viewer
  vartype ;;
```

Figure 1: A postscript viewer service: The server code registers in the name-server a high-level RPC for a postscript file viewer service (whose code is in the PostscriptViewer Objective-Caml module). The client calls the RPC when a postscript file must be viewed. The first time (init state), the server is queried, and a mobile agent carrying the postscript viewer code migrates to the client. Other invocations are immediately sent to the already received mobile agent.

```
Server  -------------------------

Definition of the player class
with two methods:
 goto: location -> unit
 play_with : player -> unit

...

let def naval_war!(l,nom)
      | wait_first!() =
   { wait_second(l,nom) }
or
   naval_war!(loc1,nom1)
 | wait_second!(loc2,nom2)=
    let p1 = new player nom1
    and p2 = new player nom2 in
    { p1#goto(loc1);
      p1#play_with(p2);}
    |
     p2#goto(loc2);
    |
     wait_first () ;;

{| wait_first () };;

Ns.register "naval_war" naval_war
  vartype;;

Client --------------------------

let war = Ns.lookup "naval_war"
           vartype in
 {|war(top_location,''my name'')}
```

Figure 2: A naval war server: The server contains the player class and a definition for high-level channel. The definition waits for the first player, then for the second player, then creates two objects that migrate to the respective player locations. At the end, the server has gone back to its initial state, and the players are autonomous.

migration of whole branches. This distributed model is embedded in the Objective-Caml language, with its full type inference and complete static type-checking with global lexical scope. All Objective-Caml programs and libraries can be compiled to be used in Jocaml programs.

The current JoCaml system contains the Objective-Caml distribution, version 1.07, with a modified compiler, a modified runtime system, the 'join' extra library and several examples of applications. Code migration is only supported for bytecode modules (modules are Objective-Caml code entities), but both bytecode modules (for migration) and native-code modules (for efficiency) can be mixed in a single executable. The JoCaml runtime also provides automatic marshaling/ unmarshaling of all data types sent on high-level channels, transparent remote-pointer creation for channels, mobile agents and objects, complete distributed garbage collection using efficient algorithms, dynamic type-checking at the name server, and partial detection of space crashes.

Future work will mainly aim at improving failure detection and security in both theoretical and practical aspects. The first examples have also shown the need for improvments in the design of language constructs.

More information on Website:
http://join.inria.fr/

■

Please contact:

Sylvain Conchon, Fabrice Le Fessant and Luc Maranget – INRIA
Tel: +33 1 3963 5801
E-mail: {sylvain.conchon, fabrice.le_fessant, luc.maranget}@inria.fr

Modelling Mobile Applications

by Stefania Gnesi and Laura Semini

There has been growing interest in wide-area distributed applications in recent years. A key concept for structuring such applications is represented by mobile agents, units of executing code that can migrate between sites.

At IEI-CNR we are currently working in the area of mobile systems. We have focused on those mobile applications that are composed of a set of components migrating among distinguished sites. In these systems there is a dynamically changing connection topology among the system sub-components. Some typical examples are: the cellular telephone system, where a Mobile Station (the telephone), moving through different geographical areas, interacts with a cell that is able to manage the communication with the final receiver; and satellite communication systems.

Most languages used to describe applications in a distributed setting provide an abstraction away from the underlying architecture, ie, from the way the application components are distributed over a network and from the way that the connections among the nodes of the network are realized. The advantages of this abstraction are independence from all physical details that are not relevant at the application level, and better reasoning on the program properties.

In particular, distributed, non-mobile languages can provide constructs, usually based on shared memory or on message passing paradigms, for the synchronous composition of the components of an application. These constructs completely abstract away from the component allocations and from the way synchronous communication is implemented. In both cases, everything should operate smoothly, ie, the behaviour of the system will be correct with respect to the definition of the semantics, since it is reasonable to hypothesise that, in general, communication failures will not occur in a distributed, non-mobile network. In fact, it can be presumed that a run-time support based on a non-mobile network will hide the few communication failures that can occur in these networks.

On the contrary, in a mobile architecture, typically based on wireless communications, the same hypothesis no longer holds and no run-time system can hide a lack of connection which can last for hours.

Two solutions can be adopted to address this problem:

1. The definition of languages to model mobile applications that include a synchronous composition construct. The drawback in this case is that, at the language level, the presence or lack of physical connection among the subsystems must be dealt with; connections cannot be assumed to be always on.

2. No synchronous communication is permitted among distinguished mobile entities, and applications are described using asynchronous languages. Ignoring synchrony, it is possible to describe most applications without mentioning any notion of connection, location, or channel, ie, keeping the description completely abstract with respect to the underlying architecture.

We have explored this solution, proposing a high level programming language based on asynchronous message passing. The language frees the developers of a mobile application from considering low level issues, and leads to descriptions that are highly readable and completely abstract from the physical architecture of the underlying net. To make our proposal effective, we have shown how to implement the language communication primitives when the underlying architecture is a mobile one.

Together with modelling issues, it is also important to define methods and tools for the formal verification of mobile systems in order to guarantee their correct behaviour. Some work has been done recently in this respect and a formal verification environment, named HAL, has been defined to check the satisfiability of safety and liveness properties of mobile systems modelled as process calculi terms. HAL was used in the formal verification of the core of the handover protocol for the GSM Public Land Mobile Network proposed by the European Telecommunication Standards Institute. The same environment has been used to verify the correctness of an implementation of the GSM Short Message Service, a service that provides an electronic postcard service working over GSM, enabling the sending and receiving of short text messages among GSM phone users. ■

Please contact:

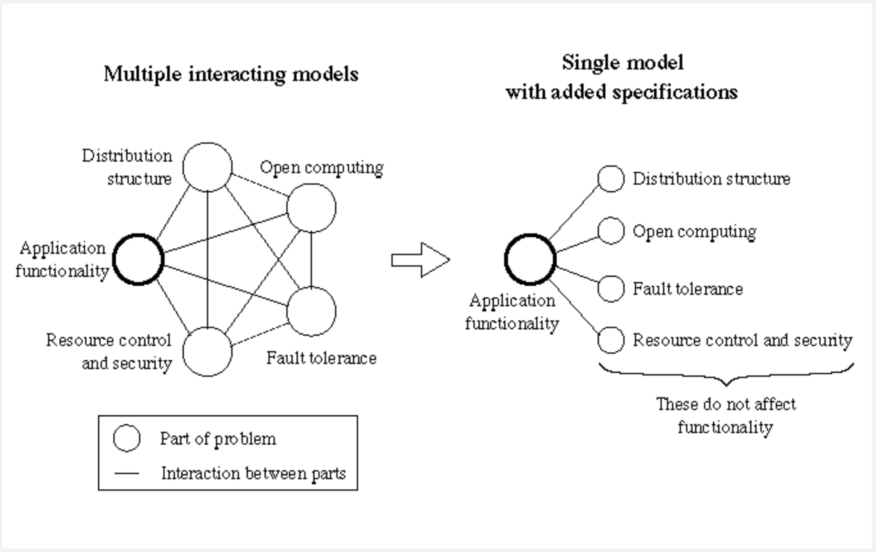
Stefania Gnesi – IEI-CNR
Tel: +39 050 593 489
E-mail: gnesi@iei.pi.cnr.it

The Mozart Platform for Distributed Application Development

by Seif Haridi, Christian Schulte and Peter Van Roy

The Mozart system for distributed application development has been released in December 1998. The system implements Oz, a concurrent object-oriented language, and provides comprehensive support for writing fault-tolerant distributed applications.

The system is the fruit of a decade of research into concurrent constraint and distributed programming. The system separates the issues of application functionality, distribution structure, fault tolerance, and open computing. The language is fully network-transparent, ie, an application obeys exactly the same semantics, independent of its distribution structure, which is specified separately from the application functionality. The system reflects distribution and fault tolerance in the language, providing abstractions that allow full control over these issues from within the language.


Multiple interacting models
Distribution structure
Open computing
Application functionality
Resource control and security
Fault tolerance
Part of problem
Interaction between parts
Single model with added specifications
Distribution structure
Open computing
Fault tolerance
Resource control and security
Application functionality
These do not affect functionality

Integration of distribution structure, open computing, fault tolerance and security into a coherent development platform.

The system has been developed as a collaboration by DFKI, SICS, and UCL, and includes a full-fledged development environment with many tools. The system also includes sophisticated constraint and logic programming abilities that are the subject of ongoing research. The Mozart system is available at http://www.ps.uni-sb.de/mozart and http://www.sics.se/mozart.

Much progress has been made in distributed computing in the areas of distribution structure, open computing, fault tolerance, and security. Yet, writing distributed applications remains difficult. This is because the programmer has to manage models of these four areas explicitly. A major challenge is to integrate the four models into a coherent development platform (see figure). Such a platform should make it possible to cleanly separate an application's functionality from the other four concerns.

The Mozart platform is a first step towards solving this problem. It is the result of three years of research into distributed programming and ten years of research into concurrent constraint programming. The current release completely separates application functionality from distribution structure, and provides primitives for fault tolerance and open computing, and partial support for security. Future releases will complete the separation for fault-tolerance and open computing, and increase support for security.

The Mozart platform implements the Oz language. Oz appears to the programmer as a concurrent object-oriented language with dataflow synchronization. Oz combines concurrent and distributed programming with logical inference, making it a unique choice for developing multi-agent systems. From a theoretical point of view, Oz is a concurrent-constraint programming language that is based on a new computation model providing a uniform foundation for higher-order functional programming, constraint logic programming, and concurrent objects with multiple inheritance.

Several research groups at the DFKI, SICS, and UCL are already developing applications in Mozart. For example, the DFKI is developing multi-agent technology in the CoMMA-MAPS project. Inside the Mozart project, we are building collaborative tools including a shared graphic editor (Transdraw), a virtual world infrastructure (Sonata), and a corpus browser for large text corpora. We have also built constraint applications in industrial scheduling, computational linguistics, and music composition. All these applications have or are reaching a substantial level of functionality. For example, the Transdraw prototype currently consists of 20,000 lines of Oz and implements a coherent graphic editor and whiteboard. Due to its transactional architecture, it has high performance even over very slow networks. It is fault-tolerant and does full and automatic remote loading of code and data.

The Mozart implementation includes an interactive programming interface based on Emacs (both GNU Emacs and XEmacs are supported), an incremental compiler, development tools (including browser, interactive constraint visualizer, parser-generator, profiler, and debugger), an Internet-wide module system with dynamic linking, persistent data structures, an object-oriented interface to Tcl/Tk, powerful interoperability features including support for sockets and a C++ interface for dynamically-linked libraries, a distributed garbage collector, and support for stand-alone applications. Furthermore, extensive libraries of constraint propagators (including global constraints for scheduling applications), distributors and search engines support the construction of intelligent applications. Performance is competitive with commercial Prolog and Lisp systems and better than emulated Java. Mozart is available for many Unix-based platforms and for Windows 95/NT.

It is interesting to compare Mozart with JDK 1.2, the current Java release. Mozart distinguishes itself from Java in five ways. First, Mozart provides true network transparency – not a single line of code has to be changed when changing the distribution structure of an application. Second, Mozart provides a truly neutral network layer – it does not make any irrevocable decisions when there are temporary or permanent faults with processes or with the network. Third, Mozart is fully extensible at run-time—one can for example upgrade the interface of a remote object interactively while the object is running and clients are communicating with it. Fourth, Mozart provides sophisticated constraint and logic programming abilities and tools. Finally, Mozart provides a much more efficient implementation of concurrency – it is literally possible to create millions of threads within a process.

Mozart is available at:
http://www.ps.uni-sb.de/mozart
and http://www.sics.se/mozart.

■

Please contact:

Seif Haridi – SICS
Tel: +46 8 633 1500
E-mail: seif@sics.se

Christian Schulte – DFKI
Tel: +49 681 302 5340
E-mail: schulte@dfki.de

Peter Van Roy – UCL
Tel: +32 10 47 83 74
E-mail: pvr@info.ucl.ac.be

Objective Caml – a General Purpose High-level Programming Language

by Xavier Leroy, Didier Rémy and Pierre Weis

Objective Caml is a general purpose programming language that combines functional, imperative, and object-oriented programming. The language is statically typed; its type system ensures the correct evaluation of programs. Types are automatically inferred. The language offers powerful constructions such as user-definable data-types, the ability to define functions by pattern-matching, and an exception mechanism. Programming in the large is facilitated by a full-fledge class-based object-oriented layer and an expressive module system.

Objective Caml belongs to the ML family of programming languages and has been implemented at INRIA Rocquencourt within the Cristal research team. Since ML's inception in the late seventies, there has been a continuous line of research at INRIA devoted to implementations and improvements of ML. Objective Caml owes a lot to the original core ML language and to our first Caml implementation (1985-1990). A new byte-coded implementation called Caml Light was developed in the early nineties. The language Caml Light is still in use, especially for education. The language was renamed Objective Caml after the incorporation of a sophisticated module system and an object-oriented layer.

As all dialects of ML, Objective Caml possesses:

- first-class functions: functions can be passed to other functions, received as arguments, or returned as results
- a powerful type system with parametric polymorphism and type inference: functions may have polymorphic types; it is possible to define a type of collections parameterized by the type of the elements, and functions operating over such collections; for instance, the sorting procedure for arrays is defined for any array, regardless of the type of its elements
- user-definable data-types and pattern matching: the user can define new recursive data-types as a combination of record and variant types; more importantly, functions over such structures can be defined by pattern matching: a generalized case statement that allows the combination of multiple tests and multiple definitions of parts of the argument in a very compact way
- exceptions for error reporting and non-local control structures
- automatic memory management.

In addition, Objective Caml features:

- a sophisticated module system: program phrases can be grouped into structures, which can be named and nested; signatures are type specifications for structures; they can be used to hide some of the structure components or abstract over some type components.

Functors, that is, functions from structures to structures, support parameterized modules

- an expressive class-based object-oriented layer that includes traditional imperative operations on objects and classes, multiple inheritance, binary methods, and functional updates.

The Objective Caml implementation comes with general purpose libraries (arbitrary precision arithmetics, multi-threading, a toolkit for graphical user interfaces, etc.) and a Unix-style programming environment including a replay debugger and a time profiler. Objective Caml programs can easily be interfaced with other languages, in particular with other C programs or libraries. The implementation is targeted towards separate compilation of stand-alone applications, although interactive use via a read-eval-print loop is also supported. Both compilation to byte-code (for portability) and to native assembly code (for performance) are supported. The native code compiler generates very efficient code, complemented by a fast, unobtrusive incremental garbage collector. The implementation runs on most Unix platforms (Linux, Digital Unix, Solaris, IRIX), under Windows 95 and NT, and on the Macintosh.

Objective Caml has been used in numerous applications involving symbolic computation (automatic theorem proving, compilation and interpretation, program analyses), and for the rapid development of applications in various areas: tools for the Web (browsers, intelligent proxies), network protocols (the Ensemble distributed communication system at Cornell, the SwitchWare active networking project at University of Pennsylvania), distributed computation, etc. The interactive system is well suited to scripting; the byte-code compiler and its dynamic linking capabilities make it possible to send or receive compiled programs from remote sites.

The Caml language is widely used for teaching in France at both undergraduate and graduate levels. It is also used in many academic projects in Europe, Japan, North and South America. Several large French corporations develop significant industrial projects in Objective Caml, including France Télécom, Dassault, and CEA (Commissariat à l'Énergie Atomique).

INRIA is setting up a Caml consortium, inspired by the World Wide Web consortium. It will offer industrial and academic partners to participate in the development, the maintenance, and the definition of new features of the language.

The Objective Caml implementation as well as extensive documentation on Caml are freely available on the Web, http://caml.inria.fr/

■

Please contact:

Xavier Leroy – INRIA
Tel: +33 1 3963 5561
E-mail: Xavier.Leroy@inria.fr

GRADE – Graphical Environment for Parallel Programming

by Péter Kacsuk and Sándor Forrai

GRADE is a graphical environment for parallel programming, based on message passing, initiated in 1994 and developed in the frame of European projects in co-operation with many partners from Hungary and abroad.

Due to the inner difficulties of parallel programming (communication, synchronisation, parallel debugging, visualisation and performance analysis), efficient program development requires powerful graphical tools. The main goal of our project is to develop a graphical environment for efficient parallel programming, based on message passing (GRADE). The developed programming tools integrated in GRADE offer both the ease of use and the flexibility to complete both simple and complex parallel applications. Using two of the most popular message-passing interfaces, the software packages PVM (Parallel Virtual Machine) and MPI (Message Passing Interface), the execution of the developed applications in a heterogeneous computing environment is allowed. The MPI (Message Passing Interface) version of GRADE has been recently completed and it is under testing. The first prototype of the GRADE was launched in 1997 and the first version for distribution is expected by the end of 1998.

GRADE is based on the following tools as main components:

- GRAPNEL – graphical parallel programming language (developed by the Laboratory of Parallel and Distributed Systems (LPDS) at SZTAKI)
- GRED – graphical editor (developed by LPDS), for parallel applications development, which supports the syntax of GRAPNEL language
- GRP2C – pre-compiler (developed at the University of Miskolc, Hungary), for C code generation (with PVM or MPI function calls) from the graphical program
- TAPE/PVM – monitoring tool for trace file generation (developed at IMAG, Grenoble, France)
- DIWIDE – distributed debugger (developed by LPDS)
- PROVE – visualisation tool for trace file analysis (developed by LPDS).

The program development cycle in GRADE is summarised as follows. In the first phase the parallel program is designed by using the graphical programming language GRAPNEL. In the next step the GRED editor creates a .GRP file from the GRAPNEL program which contains all the information necessary to restore the program graph for further development and to compile the GRAPNEL program into C and PVM/MPI code. Finally, the GRP2C pre-compiler additionally creates other auxiliary files (including makefiles used by UNIX) for building the executable file.

The C code generation and compilation is fully automated on every host of the heterogeneous cluster of workstation. Based on the executable file, the parallel program could be executed either in debugging mode or in trace mode. In debugging mode the DIWIDE distributed debugger provides commands to create breakpoints, step-by-step execution of the program, animation, etc. In trace mode a trace file is generated containing all the trace events defined by the user. These events are visualised by the PROVE, also used for performance analysis and for further optimisation.

The GRAPNEL (GRAphical Process's NEt Language) has been developed for designing distributed programs based on message passing programming paradigm where the programmer can define processes which perform the computation independently in parallel, and interact only by sending and receiving messages among themselves.

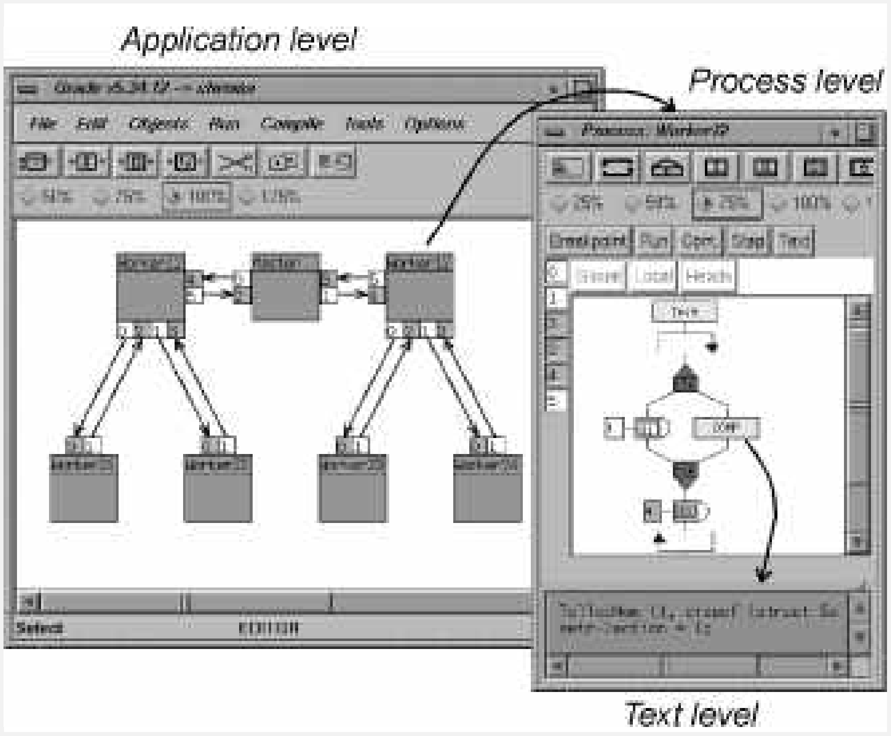
The main purpose of the graphical presentation is to give a high level of abstraction of the distributed programs where the key points are the communication operations among the processes. In addition, GRAPNEL provides the possibility of structured programming with respect to the processes as program units.

In GRAPNEL language the most fundamental unit is the process which manifests itself at two levels: Process Communication Level and Process Internal Level. Every process has two graphical views: one for describing the communication connections (ports and channels) to the other processes (Process Communication Level or Application level), and one for describing the inner structure of the process (Process Internal Level or simply Process level).

The GRAPNEL programming environment - the GRED editor - supplies the necessary support for both levels. A graphical window serves for describing the whole application at the Process Communication Level, representing the process graph of the application. Moreover, for each process a new graphical window can be opened where the inner structure of that process can be defined at the Process Internal Level of the graphical representation.

GRAPNEL supports top to bottom parallel program design, based on three hierarchical levels. Application level: the whole application is described graphically, with respect to communication among the processes. Processes, process groups, communication ports and connections among the processes are defined graphically and processes' functionalities are hidden at this level. Process level: the send and receive operations and their surrounding program structures are defined graphically. Text level: Textual program parts can be written in standard C (which can be done in any standard UNIX editor), other textual languages (eg FORTRAN) will be supported in future.

GRAPNEL can be useful for both non-professional (not experienced with distributed systems) and professional programmers. The advantages are given by the graphical environment: easy design, representation and debugging of parallel applications.



GRADE – graphical development environment.

Therefore, the main benefits of GRADE are as follows: graphical programming environment for parallel application development (program developers are not required to know the syntax of the message-passing system), automatically generated and distributed executable program code (even in heterogeneous computing environment), debugging and visualisation of the developed parallel program in a fully graphical programming environment.

As a conclusion, GRADE is a general purpose graphical programming environment for parallel programming which can be applied successfully in many application areas. As future activities, we intend to develop parallel applications in the following fields: computational fluid dynamics, numerical simulation, weather forecasting, pattern recognition and document analysis, etc. Partners from industry and academia, interested in parallel applications, are welcome.

GRADE has been installed successfully on the following hardware/software platforms: Hitachi SR2201/HI-UX (Massively Parallel Processor), SGI/IRIX 5, SGI Origin Series/IRIX 6, Sun Solaris/

SPARC 2.6, Intel 2.0/Linux. Currently, GRADE is used by the following institutes: University of Westminster, London; Institute of Computer Systems, Slovak Academy of Sciences; Universidade Nova de Lisboa; Universidad Autonoma de Barcelona; Technical University of Gdansk, Poland; University of Miskolc, Hungary; University of Vienna; Polish-Japanese Institute of Computer Techniques.

Up-to-date information about GRADE and the Laboratory of Parallel and Distributed Systems of SZTAKI is available at http://www.lpds.sztaki.hu/. ■

Please contact:

Péter Kacsuk – SZTAKI
Tel: +36 1 329 7864
E-mail: kacsuk@sztaki.hu

Sándor Forrai – SZTAKI
Tel: +36 1 329 7864
E-mail: forrai@sztaki.hu

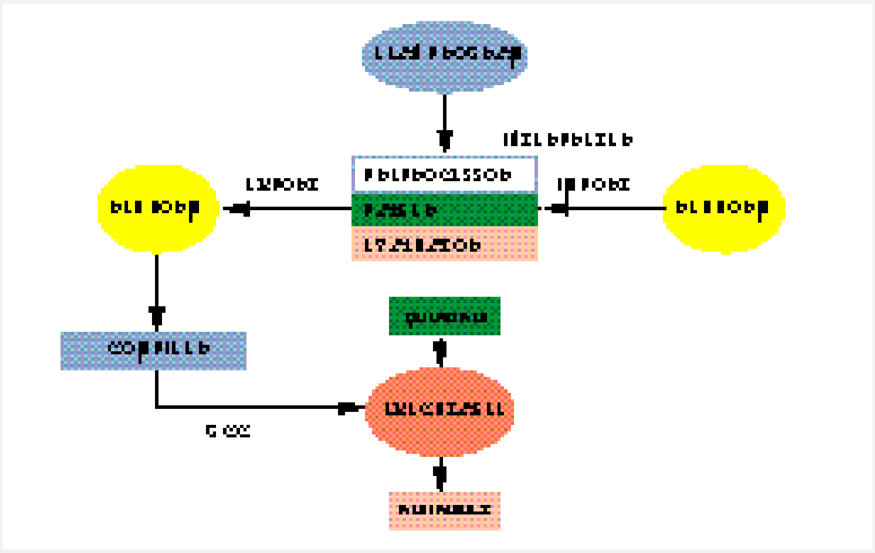
Programming with Rewrite Rules and Strategies

by Hélène Kirchner

ELAN is a language to express non-deterministic computations via rewrite rules and strategies. Strategies are the part of the program that specifies the way rule application is to be controlled. Compilation techniques are studied in this context for an efficient execution of these programs. ELAN is developed in the PROTHEO project common to CNRS, INRIA and Universities of Nancy.

Rules are present in many domains of Computer Science: let us mention for instance production rules, inference rules, grammar rules, transition rules, constraint simplification rules, program transformation rules, to cite a few. All of them are actually rewrite rules, ie schemas allowing to transform expressions. A set of rewrite rules is a program (or a theory) in rewriting logic. To execute such a program with a given query (the expression to rewrite), at each step one has to choose the rule to apply and the position in the expression where the rule is to be applied. This choice may be complex, even in a case that may look simple of simplification of arithmetic expressions.

Due to this inherent non-determinism, controlling rewrite rule application is an important issue for all kinds of rules. Control is expressed as search plans in theorem provers; action plans in scheduling; tactics or tacticals in logical frameworks; lazy evaluation in functional programming; or via constructs like cut in logic programming. In most programming languages the evaluation strategy is fixed, which avoids non-determinism, so that the evaluation process is then easier to implement. The counterpart of this option is that programs are dependent on this built-in evaluation strategy. Any control deviating from this has to be encoded via data or program structures.


General architecture of the ELAN system.

The approach followed in the ELAN project is different. Programs are composed of rewrite rules and strategies whose purpose is to guide choices in non-deterministic situations, to select rules to be applied, or to cut useless branches in the computation tree. Strategies are terms with higher-order types that are applied to first-order terms.

Since rewriting is inherently non-deterministic, in ELAN, a computation may have several results. This aspect is taken into account by a backtracking capability, and choice strategy constructors (don't know, don't care, don't care one) that allow specifying whether a strategy call returns several, at-least one, or only one result. Strategies can be sequentially composed and iterated. Elementary strategies are labelled rules. From them and the strategy constructors, more complex strategies can be expressed. In addition, the programmer can declare new strategy operators, define them by rewrite rules, and design rules to simplify strategy expressions.

The current version of ELAN includes an interpreter and a compiler written in C++ and JAVA. Both can interact via an exchange format (REF, for Reduced ELAN Format) which is a term representation of ELAN programs. This format appears to be a convenient way for to transform ELAN programs making use of ELAN itself, and is the key for the implementation of a reflection mechanism in ELAN.

Initially, the system was designed for specifying and prototyping theorem provers, constraint solvers and decision procedures, and for studying their combination. For instance COLETTE is a solver for constraint satisfaction problems designed in ELAN, where propagation and consistency techniques can be experimented with flexible definitions of strategies.

To deal with such applications, efficiency is crucial and motivates the development of the ELAN compiler. Compilation techniques for non-deterministic rewrite programs with strategies are based on efficient data structures such as matching automata and compact bipartite graphs, but also on a careful memory and backtracking management. For implementation of backtracking, two functions are usually required: the first one, to create a choice point and save the execution environment; the second one, to backtrack to the last created choice point and restore the saved environment. These functions are implemented in assembly language and may be useful in other contexts, for instance to implement backtracking in imperative languages.

A determinism analysis is performed in the ELAN compiler, where every rule and strategy in the program is annotated with its determinism mode which classifies the strategies according to the number of their expected results. In case of deterministic strategies with 0 or 1 result for instance, many choice points are dropped at run time. This approach considerably reduces the search space size, the memory usage, the number of necessary choice points, and the time spent in backtracking and memory management. The determinism analysis simply makes possible to run programs that could not be executed without it due to memory explosion. This was the case in particular for constraint satisfaction problems. In other examples, this analysis significantly decreased the number of set choice points and improved the performance. The ELAN compiler can handle large specifications (thousands of rules and strategies), long computations (more than 20.10^9 applied rules), and can apply on some examples up to 17.10^6 rewrite steps per second.

The simple rewriting paradigm that is implemented in ELAN as the evaluation mechanism of the language actually provides a logical framework in which deduction systems can be expressed and combined. Rewriting logic is a natural semantic framework for concurrency and parallel programming, and for the specification of systems and languages. It has also good properties as a logical framework for representing logics. A growing number of research efforts exploring the application of rewriting logic in all these directions are being carried out worldwide, and several languages based on rewriting logic are being designed and implemented. ELAN is one of them and can be compared to other systems such as ASF+SDF (CWI, The Netherlands), Maude (SRI, California) or Cafe-OBJ (JAIST, Japan). The originality of ELAN with respect to these other languages is to provide non-deterministic computations and to have initiated the implementation of strategies.

Perspectives for the ELAN project include improvement of the overall design of the system, the study of reflection properties of the framework, and the design of proof editing and debugging tools.

For more information on the ELAN system, see http://www.loria.fr/ELAN/ ■

Please Contact:

Hélène Kirchner – LORIA
Tel: +33 3 83 59 30 12
E-mail: Helene.Kirchner@loria.fr

Using Co-ordination to Parallelize Existing Sequential Programs

by Farhad Arbab, Kees Everaars and Barry Koren

The co-ordination language MANIFOLD, developed during the 1990s at CWI, has important applications in the parallelization of computation intensive sequential programs. The language is based on a novel model for control-oriented coordination (IWIM).

Programming in MANIFOLD (see ERCIM News 35, page 33) is a game of dynamically creating process instances and (re)connecting the ports of some processes via streams (asynchronous channels), in reaction to observed event occurrences. This style reflects the way one programmer might discuss his interprocess communication application with another programmer by telephone (let process *a* connect process *b* with process *c* so that *c* can get its input; when process *b* receives event *e*, broadcast by process *c*, react on that by doing this and that; etc.).

As is already clear from this phone call, processes in MANIFOLD do not explicitly send a message to or receive a message from another process. Processes in MANIFOLD are treated as black-box workers that can only read or write through the openings (called ports) in their own bounding walls. Always a third party – a coordinator process called 'manager', – is responsible for setting up the communication channel between the output port of one process and the input port of another process, so that data can flow through it.

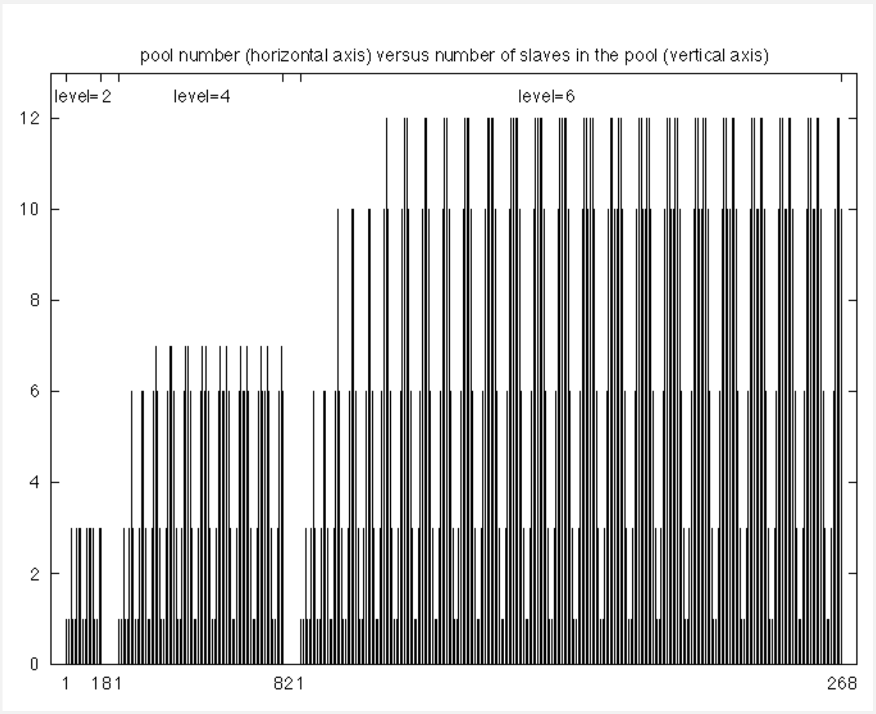
This setting up of the communication from the *outside* is very typical for

MANIFOLD and has several advantages. An important advantage is that it results in a clear separation between the modules responsible for computation and those responsible for co-ordination, and therefore also strengthens the modularity and enhances the re-usability of both types of modules.

MANIFOLD has been successfully used at CWI to restructure a sequential existing implementation of a real-life heavy-duty Computational Fluid Dynamics application (with a semi-coarsened multi-grid Euler solver algorithm). The sequential version of this software was created in the framework of contract research financed by the European Union (BRITE/EURAM). For the restructuring of the sequential application into a parallel application we used a master/worker protocol implemented in MANIFOLD.

The idea is simple. In a coordinator process we create and activate a master process that embodies the computations of the main program of the sequential version. When we arrive in the 'master', at some work that could be parallelized (in our case the pre- or post-relaxation on the different grids), the master does not perform these computations itself but delegates it to a number of 'worker', processes. Each time the master needs a worker to delegate some work to, it raises an event to signal the coordinator to create a worker. In this way a pool of workers is working for the master, each worker performing pre- or post-relaxation.

When the workers have finished their relaxations (this creates a synchronization point in the application), the master proceeds its sequential work until it again arrives at a point where it wants to use a pool of workers to delegate the relaxations to. The Figure shows the dynamically created pools of workers during a semi-sparse multigrid run. Here we can see for example that for level 6 there are 268 pools of workers created, in which a total of 1838 (ie, the total of the heights in this histogram) workers did their relaxation work. In the sparse-grid-of-grids approach this simple master/worker implemented in MANIFOLD was able to improve the execution time from almost 9 to over 2 hours on a 4-processor machine.



Some two thousand odd processes, co-ordinated by MANIFOLD, compute an aircraft wing. Horizontal: the number of 'worker-pools' created during the parallel applications. Vertical: the number of 'workers' in the 'pool'.

The modularity of MANIFOLD also enables the introduction of concurrency step by step. We can therefore proceed as follows. We initially plug a block of code as a monolithic computing process into a concurrent structure, in order to obtain a running parallel/distributed application. As more experience is gained through running the new application, computation bottlenecks may be identified. This may lead to replacing some such monolithic blocks of code with more MANIFOLD modules that coordinate the activity of smaller blocks of computation code, in a new concurrent sub-structure.

The MANIFOLD system runs on multiple platforms and consists of a compiler, a run-time system library, a number of utility programs, and libraries of built-in and predefined processes of general interest. Presently, it runs on IBM RS6000 AIX, IBM SP1/2, Solaris, Linux, Cray, and SGI IRIX.

For more information, see website http://dbs.cwi.nl/cwwwi/owa/cwwwi.print_projects?ID=57, and ID=58.

■

Please contact:

Farhad Arbab, Kees Everaars (MANIFOLD) and Barry Koren (CFD) – CWI
Tel: +31 20 592 4056
E-mail: {Farhad.Arbab, Kees.Everaars, Barry.Koren}@cwi.nl

EINS-Web: User Interface Evaluation in Digital Libraries

by Silvana Mangiaracina and Pier Giorgio Marchetti

The construction of new user interface paradigms, or metaphors, in the interaction of humans with huge collections of information is currently a hot topic in both HCI and Digital Libraries (DL). We describe the EINS-Web user interface, designed and evaluated in the frame of the BRIDGE and CIME projects, co-financed by the European Union and coordinated by the European Space Agency and the EINS consortium (European Information Network Services). The Library of the Italian National Research Council (CNR) in Bologna was selected as the test site for the evaluation of EINS-Web.

The EINS-Web interface enables users to access distributed collections of bibliographic and textual databases, and provides a seamless interaction with the World Wide Web. The interface design has been guided by a heuristic evaluation, using a spiral design approach. This methodology was adopted as it is largely software-independent and proactive. It thus makes it relatively easy to incorporate suggested adaptations during the design and testing process.

The Design-Evaluation Spiral Process

The Library of the Italian National Research Council (CNR) in Bologna was selected as the test site for the evaluation of EINS-Web. The evaluation team consisted of a number of researchers from the CNR campus with their information problems in a varied set of disciplines, such as chemistry, material science, electronics, physics, geology, environment, etc., plus a mixed group of evaluators consisting of information specialists and user interface experts.

In the construction of the EINS-Web interface, we reused the design efforts employed in the development of the previous version of the interface (BRAQUE PC where BRAQUE = BRowse And QUEry), developed for the Windows environment.

A heuristic evaluation method then assessed user satisfaction, ease of learning, ease of use, error prevention and efficiency of the interface and was used as a feedback tool to drive a spiral design process. The evaluation began in April 1996, when BRAQUE 1.2 was released. Problems identified by our set of users were analysed against the above heuristic criteria. A number of issues were signalled, mainly of an aesthetic nature. Some were serious, heavily influencing the user interaction. Where possible, suitable solutions were proposed. As the results of the BRAQUE PC evaluation were greater than expected, it was decided to assess and improve the methodology in the design of the next generation Web interface. This new version was called EINS-Web, as the international EINS consortium had decided to adopt and test this interface.

All the information collected during the BRAQUE evaluation sessions, such as evaluators' opinions and implementers' replies, was used as input to drive the design process for the EINS-Web interface. Two different evaluation sessions were conducted: one involving a mixed group of experts (evaluators) and users; one with experts only. A new heuristic evaluation form was used by the evaluators: for each 'usability' problem, a rating value was assigned – we agreed to assign values from 1 (the interface does not take this problem into consideration at all) to 5 (this problem has been completely solved).

The evaluators were requested to identify potential usability problems and to link each problem to the specific heuristic it violated. Multiple heuristics could be linked to any given violation. In the experts-only evaluation session, the same evaluation - covering the same information problems - was run by four different groups of evaluators.

Assessment of the evaluation methodology and lessons learned

The evaluation was useful in detecting certain design issues that had been neglected to some extent. In particular two problem areas were identified. The first related to feedback and visibility of system status. The second regarded user background knowledge and the user conceptual model. When the results of the four different groups of evaluators were compared, it could be seen that they had used all the heuristics present in the evaluation form, either in positive matches (ie a score >= 3), or in negative matches (ie one or more problems recognized were linked to the heuristic). This result shows that all the pre-selected heuristics were relevant. We noticed slight differences in the evaluation results, depending on the presence of real users or not. It appeared in fact that in a 'pure' heuristic evaluation session (only interface experts, no real users) it was possible to detect problems relating more to the interactive behaviour of the interface, such as users' behaviour problems, conceptual user model, aesthetic design. The evaluation with real users made it possible to examine concrete, real-world information seeking interaction problems. Matching between interface design and user expectation is difficult when information space is dispersed over very large collections: expert users want to increase interface functionality to achieve their goals, whilst non-expert users want to reduce interface functionality in favour of intuitive and simple features.

As a preliminary conclusion we feel that the interaction among evaluators, implementers and designers has contributed significantly to the success of the spiral design methodology, and is very necessary to cope with the requirements of designing interfaces targeted at the rapidly evolving Internet world.

■

Please contact:

Silvana Mangiaracina – CNR, Bologna
Tel: +39 051 639 8026
E-mail: mangiaracina@area.bo.cnr.it

Service Creation for All: Initial Results of the TOSCA Project

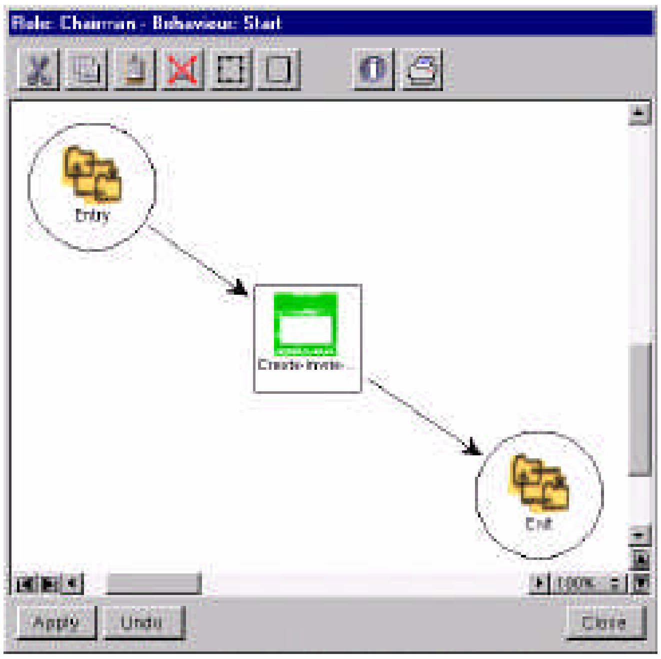
by Richard Sinnott

The TOSCA (TINA Open Service Creation Architecture) project aims to develop an open service creation environment that can be used to expedite the creation and subsequent validation of multimedia-based telecommunication services. In particular, the project is attempting to widen the audience of 'would-be' service creators. This article gives an overview of how this is being achieved in TOSCA.

The TOSCA project which began in September 1996, has recognised the need to differentiate between the different creators of telecommunication services that might exist in the modern marketplace. For example, service designers as might be employed by public network operators are likely to create services using a detailed knowledge of the expected service behaviour. Business consultants or telecommunication managers on the other hand, are more likely to create services based on an understanding of the end user requirements, as opposed to a detailed knowledge of the service specification and low-level service construction. Regarding this latter point, a lack of understanding of technologies such as C++, SDL (Specification and Description Language), or distributed platform technologies such as Orbix etc. should not be prohibitive to the service creation process.

As well as providing different levels of abstraction at which services might be developed, a key factor on the success of a service creation environment is the speed at which a multitude of services can be created. This includes integration with their service surround as might be the case in the usage of an existing accounting service say, by a newly created tele-learning service.

To accommodate these requirements on rapid service creation and the potentially wide spectrum of would-be users, TOSCA has taken an approach based upon object-oriented frameworks in accompaniment with graphical (and intuitive) service design tools - so called Paradigm tools. A framework may be considered as a mostly complete model of a service with some pre-defined flexibility points where the developer can intervene (either manually or through a paradigm tool) to specialise the behaviour of the service.



Example of the kind of user application that might be generated for a chairman from the framework using the paradigm tool.

Currently frameworks in C++/Orbix and SDL have been completed. These were based around an existing implementation of a TINA based multimedia conferencing service (TIMMAP). Both of these frameworks started from a TINA ODL (Office Document Language) and IDL (Interface Definition Language) description accompanied by informal behavioural descriptions and use cases. We note that tool support (Y.SCE) was used to convert these ODL/IDL descriptions to SDL stubs and skeletons whose behaviour was added (and checked) using the SDL Design Tool of the Telelogic TAU toolset.

Different paradigm abstractions have been considered in TOSCA. Two of those which resulted in prototype implementations were:

- functional block paradigm: here the design of the service is based on architecting a collection of building blocks representing particular functional decompositions of the overall service behaviour
- movies paradigm: here the design of the service is based on considering the service behaviour as a whole through a sequence of snapshots of its expected functionality.

The following highlights how the functional block paradigm tool (Cadenza) is used to create the different roles that might exist in a multimedia conference.

Specifying the Roles in a Multimedia Conference

The functionality associated with these roles is achieved through inserting behaviour into specific framework flexibility points. For example, enabling a chairman to invite other participants (the default functionality is that the party may terminate or suspend their participation in a session) is achieved through specialising the framework flexibility point related to starting user sessions so that a chairman has an invite window created. Other flexibility points related to stopping, suspending or resuming user sessions and starting, stopping, suspending and resuming or service sessions have also been identified.

Specialising the Start-Up of the Chairman Role

More complex building block scenarios are also possible. The output of these tools is the specialising C++ or SDL necessary to complete the service implementation or SDL model from the respective frameworks. An example of the kind of (service-session) user application (for interacting with the service implementation and the service model once a service session has started) that might be generated for a chairman from the framework using the paradigm tool is shown in the figure.

Example of Resultant Chairman Control Window

Once the SDL model of the service is complete, it is used as a basis for deriving test cases to run against the C++/Orbix implementation of that service. Tool support (a CORBA/TTCN gateway; Common Object Request Broker Architecture/Tree and Tabular Combined Notation) has also been developed to allow for these test cases to be executed directly against the service implementation.

The TOSCA project is funded under ACTS (Advanced Communications Technologies and Services). Further information on the TOSCA work can be found at: http://www.teltec.dcu.ie/tosca ■

Please contact:

Richard Sinnott – GMD
Tel: +49 30 3463 7298
E-mail: sinnott@fokus.gmd.de

Pushing Environmental Information

by Tuula Käpylä

VTT Information Technology has been designing and implementing a push application for EIONET (European Environment Information and Observation Network) to intensify the acquisition and distribution of environmental information of the European Environment Agency (EEA).

Information overload makes it more and more difficult to get the right information at the right time. The ‘traditional’ pull technology model is becoming a limit for more efficient use of the WWW, specifically requesting information from a particular source, eg downloading a page with a browser is an example of pull technology. At present, pull technology seems to be adequate for information that is not time or content critical. As the WWW grows in size and complexity, however, new information delivery models such as push technology will become increasingly important. Push technology is a technology by which a program running on a workstation can either request or receive information from the WWW automatically (on a pre-arranged schedule or when certain events occur) and then display that information on the screen.

Personalised Information flow by IDA-PUSH

EIONET is a co-operative organisational network of institutions that assists the European Environment Agency in providing the European Community and its member states with environmental

Push Advantages

- Push technology can reduce the burden of acquiring data for tasks in which there is a large information flow. Push technologies improve efficiency by downloading information to a user's system in a scheduled fashion so that it can be rapidly viewed, thereby eliminating the risk of the user never viewing the updated information. The user always has the latest information. No longer do users have to search for the information.
- Push technology can reduce the burden of acquiring data for tasks where occasional, time-critical data must receive immediate attention.
- Businesses are able to target users with more precision, focusing on those who are more likely to benefit from their products or services.
- Automatic downloading of software upgrades and fixes is a way to deliver software faster and, at the same time, reduce the costs associated with packaging and selling through the retail channel. A key factor in allowing such distributed services is a security system applicable on either side of a firewall.
- Only new and changed information has to be sent to the computer, so access to the Internet and download time is minimised.
- The software is run on the client side, minimising processor use of company’s WWW servers. Servers can use more processor time for data production rather than the processing of numerous client requests and the transmission of much data over the network. Servers can better manage the amount of data transferred over the network.
- Response time is generally quicker because the information is on a local computer, not on a remote server.
- Because push applications run mostly on the client side, users can more easily protect their privacy. In many push applications the user profile and the log information about the user's behaviour are stored in the user's computer. An ordinary WWW application stores this data in the content provider’s database.
- Push technology enables intelligent information filtering based on personalised user profiles describing required information needs.
- When some data must be provided to employees in compliance with laws, company rules, health and safety and quality control, push technology can help here if combined with some mechanism for reporting when users have spent sufficient time assimilating the received information.

information. It also allows the electronic exchange of information between these organisations. EIONET is currently an intranet connecting multiple national hosts (intranet nodes).

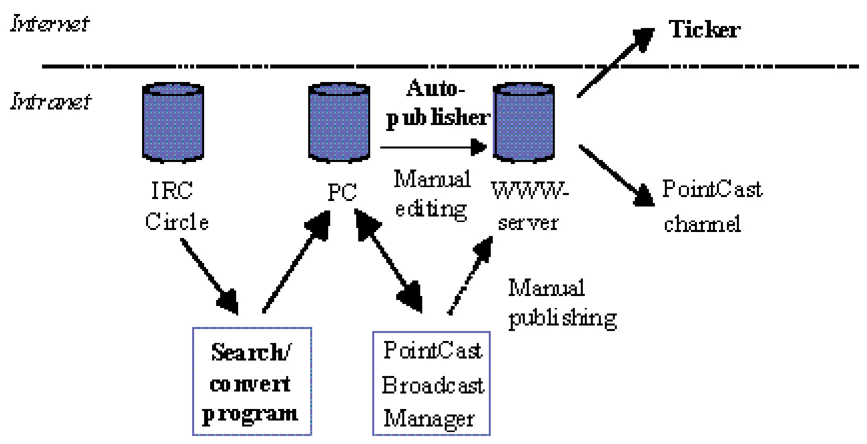
The IDA-PUSH application manages documents retained and edited under a workgroup program (IRC Circle) based WWW repository. This workgroup program enables interest group members to load documents and modify them on the server.

The push functionality means the ability of the IDA-PUSH application to inform its users of the changes in the WWW repository that might interest the user. The information mediated to the user by push technology can consist of actual data or links to the data. Users can personalise their information flow by selecting which channels they receive as well as the type of information broadcast within each channel.

The information coming to a user's workstation is a summary of a larger document that the user can access by requesting additional details. By clicking on a headline the full article appears on the screen. Personalised screen savers keep the user informed of the latest information with headlines that scroll across the screen. The IDA-PUSH screen saver uses headlines to inform users of new meetings and the latest documents. A personalised ticker is a movable, bar-shaped window that can even display time-sensitive information while the user works in other applications. A ticker can also run on the screen saver. IDA-PUSH uses the PointCast Intranet Broadcast Manager to broadcast information.

The IDA-PUSH application is general enough to be applied to different WWW repositories with a reasonable amount of work. In the IDA-PUSH project we have also reviewed the current market availability of push tools as well as the differences between them. The project was carried out in co-operation with EEA and Tieto Corp.

■



The server-side architecture of the IDA-PUSH application.

Please contact:

Tuula Käpylä – VTT
Tel: +358 9 456 6054
E-mail: tuula.kapyla@vtt.fi

NESSIE – Awareness Environment

by Wolfgang Prinz

The NESSIE project develops an infrastructure through which the events in an electronic environment can be captured, communicated and presented across the Internet.

The support of awareness is an important requirement for the successful implementation of groupware applications, and for the use of the Internet as a social medium. Awareness involves the perception of important events and relevant actions of remote partners engaged in synchronous or asynchronous cooperation. It is important that the notification occurs in social and task-related contexts and that longer periods of absence between notifications can be bridged, eg, through summary reports or multimedia process visualizations.

The goal of NESSIE is the development of an infrastructure through which the events in an electronic environment can be captured and made available to authorized users across the Internet. It should be possible to signal all kinds of events, in particular, presence, actions and movements of other participants, agents, or objects. The NESSIE System is not bound to a specific application, but instead offers its services globally, across applications.

Architecture

The NESSIE system consists of a server, which uses an open protocol to receive event information from any type of application. Through a client, users – or other programs – can register their requests for event information with the NESSIE server. Additionally, the type and conditions of the indicators can be individually configured. This serves as the technical basis for reciprocal perception of events and for group awareness in a Social Web.

NESSIE can also be used to set up a personal information service that monitors and reports the events in an electronic environment. Events might range from learning who is visiting one's homepage to an individually configured Push-Service, which would announce a traffic jam as one heads home in the evening.

Tangible indicator: Nessie lets the balloon pop out of the box.

Presentation Form

NESSIE uses various representations to show events, eg PopUp windows, animations, ticker tapes, or sound. Beyond electronic representation, the NESSIE project experiments with tangible interfaces which support presentation of events independent of a computer screen.

Three-dimensional virtual environments provide a new medium for the presentation of events. For this purpose, NESSIE uses the SmallView system developed at the GMD Institute for Applied Information Technology to establish a connection between real and synthetic virtual environments. SmallView, which is based on the Internet standard VRML (Virtual Reality Modeling Language) for the visualization of three-dimensional scenes, lets several users communicate and interact across the Internet within a distributed virtual environment. Gestures and actions of the user may be captured by sensors such as MOVY, a prototype developed in GMD Institute for Applied Information Technology, which can sense acceleration, determine rotation and movement of an object and radio the information to a local computer system.

Partner in the NESSIE project is the StatOil Research Center, Norway. NESSIE is part of our research program The Social Web: New Forms of Interaction in Virtual Environments (http://orgwis.gmd.de/projects/SocialWeb). ■

Please contact:

Wolfgang Prinz – GMD
Tel: +49 2241 14 2730
E-mail: wolfgang.prinz@gmd.de

Eurosearch: a Federation of European Search Engines

by Martin Braschler, Mounia Lalmas, Luigi Madella and Carol Peters

The objective of the EuroSearch project is to build a federation of European search engines. The main aims of the federation are to join forces in order to be able to better compete with global search engines, to enhance the visibility of European web sites, and to help to preserve European language and cultural diversity. The technologies under development will provide linguistic support for querying over different search services and enable the automatic generation of catalogues, best reflecting local cultures within the federation.

The decision of the EuroSearch consortium of industrial (Italia On-Line, Pisa; CINET, Barcelona; EuroSpider Information Technology, Zurich) and academic partners (CNR, Pisa, and Dortmund University) to build a federation of European search engines originated from the consideration that the World Wide Web is still dominated by US culture and so far little effort has been put into promoting European web sites. A study of the incoming traffic of the services provided by the EuroSearch partners determined that about 70% comes from the same country or from countries using the same language; of the outgoing traffic more than 50% is directed to the US, while almost all the rest remains in the country of origin. This situation has been analysed as depending mainly on language barriers between European countries, the poor multilingual support in traditional search engines, and on the US cultural domination of most popular web catalogues.

The EuroSearch project thus aims at helping to restore linguistic and cultural equilibrium on the Web by building a pan-European federation of national search and categorization services. The main objectives of the federation are to:

- promote traffic across Europe by exchanging links and sharing services
- provide language support for query translation
- provide tools for automatic categorization in order to overcome the high costs of traditional catalogues, still affordable only by big international organisations.

The Cross-Language Approach

The aim of the EuroSearch distributed, multilingual service is to permit users to enter queries in their own, or their preferred language, and to carry out search and information retrieval over some or all of the federation's national sites.

Differences in the partners' document collections and indexing mechanisms have led to the implementation of different search strategies, depending on the collection to be queried. The cross-language search component of EuroSearch thus activates two distinct types of searching:

- Query translation using a multilingual lexicon; this employs the pivot language concept and semantic indicators are assigned to polysemous words to permit interactive sense disambiguation. Queries can also be expanded using corpus-extracted data.

- Similarity thesaurus technology; a multilingual similarity thesaurus contains entries linking terms in one language to a list of similar terms in another, each assigned with a similarity value based on statistical co-occurrence, ie basically how often the terms co-occur in similar texts taken from training data.

The languages covered are currently German, Italian and Spanish, plus English. The two approaches are integrated through the development of common translation server interfaces and data exchange formats; this will facilitate future extensions of the Eurosearch components.

A preliminary simplified prototype of a Translation Server has been developed and integrated in the Arianna search engine, allowing queries in Italian to be formulated and directed to Alta Vista. This server will be extended with the addition of a corpus-based query expansion mechanism. In 1999 the integration of the linguistic resources on all the federated services will be completed.

The Automatic Categorization Technology

Another important goal of the project is to facilitate the creation of Web catalogues by developing techniques for the automatic categorization of documents. In this way, even small corporations will be able to develop their own catalogues.

The categorization approach is grounded on an automatic textual analysis of web documents associating weighted terms with documents. The determination of the weighted terms is based on the description-oriented indexing approach developed at the University of Dortmund. It takes into account features:

- specific to web documents (whether a term appears in a title, a heading, or is highlighted)
- standard to text documents (term frequency).

The weights are probabilistically determined using the Least Square Polynomial (LSP) approach and a test-bed of pre-categorized documents taken from the 'Computers and Internet' part of the Yahoo! catalogue.

This approach produces two main results:

- the automatic classification of new documents into appropriate categories
- the determination of documents that belong to given categories.

The approach is fully automatic, and is portable to the various languages involved in the federation. We are currently applying our techniques to German web documents from the DINO-online catalogue.

A preliminary on-line prototype is now running, and an engineered version is available in the Arianna catalog. This is one of the first examples in the world of automatically generated catalogues available on the Web.

For further information and demos, see the EuroSearch Web site at: http://eurosearch.iol.it/

■

Please contact:

Luigi Madella – Italia Online
Tel: +39 050 944258
E-mail: l.madella@pisa.iol.it

APEX - Awareness and Promotion EXercise

by Mike Ashworth and David Emerson

APEX is a two-year project funded by the European Union ESPRIT IV programme. Its aim has been to promote the benefits of using High Performance Computing (HPC) in combination with Computational Fluid Dynamics (CFD) to support the next generation approach to industrial research and development in the process industries, with emphasis on the use of Information Technology for simulation and prediction. A multi-media CD-ROM describing the benefits of CFD and HPC has been widely and freely distributed throughout the European Community.

The very fast pace of change within HPC hardware and continuous developments of CFD software require a way of conveying information very rapidly. A specific goal of the APEX project was to develop a multimedia CD-ROM that could be distributed to relevant industrial companies. The CD contains detailed information about HPC and CFD and results from two industrial applications from the ESPRIT III HP-PIPES project (#8114). An additional important feature is a Cost Benefit Analysis (CBA) section that will allow senior management to consider the investment decisions and risks concerning such technology. The partners involved in APEX are:

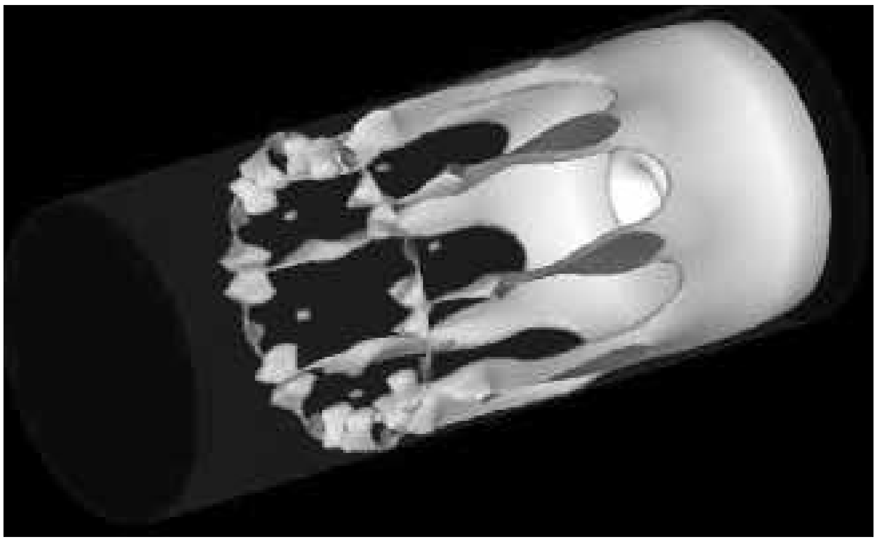
- CLRC Daresbury Laboratory, Warrington, UK, responsible for the HPC section and industrial application of Tioxide reactor
- Instituto Superior Tecnico, Lisbon, Portugal, responsible for the CFD section and industrial application of EDP power station burner
- Paras Ltd., Isle of Wight, UK, responsible for the CBA section and project management.

The APEX CD highlights two industrial applications from the process industries. In collaboration with industrial partners, the HP-PIPES project developed a CFD code to solve the 3D incompressible Navier-Stokes equations and to augment the core code with modules describing the physics and chemistry of particular applications. It was demonstrated that advanced numerical simulation can be used as a realistic predictive tool for the design and operation of complex chemically reacting flows and combustion systems. The core code, developed at Daresbury Laboratory, describes fluid flow and heat transfer in process engineering systems and has been designed to exploit the power of massively parallel high performance computing systems. It has been written in a portable and modular form to facilitate the future development of modules for the specific features of particular applications.

An Industrial Application Example: Production of Titanium Dioxide

Tioxide plc, formerly owned by ICI, is one of the world's largest producers of the white pigment titanium dioxide that is widely used in materials such as paints because of its good light scattering characteristics. The critical parameters of the product are the mean size of the particles, which dominates its scattering power, and the distribution of the different particle sizes about the mean diameter, which affects the range of wavelengths that are scattered and the resulting effectiveness of the pigment in materials such as paints.

The HP-PIPES code was designed to overcome limitations in the existing model of the titanium dioxide process, in particular in the areas of the physical and chemical modelling and in the resolution of the flow field. At the start of the project axi-symmetric simulations were taking approximately one month to converge. The parallel implementation reduced simulation times to a few hours. This opened up the possibilities of exploring in more detail the phase space of physical and chemical parameters embedded in the model, enabling improvements in the theoretical bases of the various physical and chemical sub-models and moving the models to three spatial dimensions better to simulate real reactor geometries. The ultimate aim of the HP-PIPES project was to develop a predictive model capable of determining the optimum reactor configuration and operating conditions so as to produce particles of a specified mean diameter and standard deviation.


Results of the HP-PIPES Tioxide reactor simulation.

The Titanium Dioxide chloride process occurs in a basic pipe reactor. The oxidation process involves several complicated stages which may include the nucleation, growth and coagulation of the titania particles and the dissociation and recombination of chlorine produced in the reaction. In general the reactors have multiple inlets through which the feed materials, titanium tetrachloride and molecular oxygen, are fed at various temperatures, velocities and mixture fractions. The products, and any unreacted input materials, flow through the reactor at high speed. Initially, titania particles are assumed to form as nuclei and these become larger through surface deposition and coagulate as they flow through the reactor. These various processes produce a continuous size distribution of titania particles with a broad distribution of sizes.

The particle growth process is modelled by discretising the size distribution in terms of a fixed number of size intervals characterised by a mean diameter. Each of the size classes is then treated as a distinct transported variable in its own right The models can employ anything up to 100 class sizes for high accuracy computations. The calculations of the various source terms describing the nucleation, growth and coagulation is extremely time consuming as the coagulation terms fully couple all the size class variables in each finite volume. It is obvious that three-dimensional computations for systems with this number of class size variables would be completely impractical and uneconomic in the absence of parallel systems.

The APEX CD-ROM

The APEX CD-ROM is an interactive multi-media presentation, which introduces CFD and HPC, shows results from two typical applications (a chemical reactor and a power station boiler) and guides the viewer through the investment decision process.

The CD-ROM is available cost-free from the contact address below or by filling in the electronic form on our Web Page http://www.dci.clrc.ac.uk/Activity/APEX ■

Please contact:

Mike Ashworth – CLRC Daresbury Laboratory
Tel: +44 1925 60 3663
E-mail: M.Ashworth@dl.ac.uk

MOVY – Wireless Sensor for Gestures, Rotation and Movement

by Peter Henne

Increasingly, innovative applications need to be able to sense spontaneous (natural) movements of the user. This need is obvious for a head-mounted display (data helmet) that lets the user look and wander around in a virtual 3-D world. More recently, the idea to capture and communicate natural gestures to manage net-based teleconferencing applications has defined a new set of requirements for motion sensors (trackers). MOVY is a wireless input device that captures the user's gestures, movements of the hands or the head, shifts in the focus of vision or any unspecific movement that a program may be able to interpret.

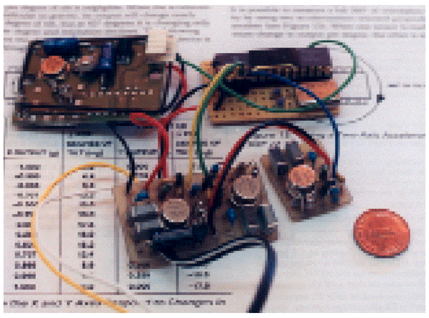
Magnetic trackers are widely used today. This technology uses a strong artificial magnetic field. The sensors that move about in this magnetic field can capture the variations in its direction and strength. They determine position and movement of the tracker on that basis. Besides their high price, magnetic trackers have some technical shortcomings: it takes considerable time and effort to install and to calibrate them. The natural magnetic field and even metal objects may confuse the sensors, the space where movement can be captured is rather limited.

Requirements for the next generation of trackers would include:

- the tracking technology does not rely on any artificial external sources
- the tracker can use wireless data transmission to get the data into the computer, so that no cabling hampers spontaneous and natural gesturing
- the tracking device proper should be small enough to fit comfortably in the user's hand or to be worn like a ring on the index finger
- the tracker should, above all, be easy and cheap to produce.

MOVY

At the GMD Institute for Applied Information Technology, the MOVY inertial tracker has been developed to meet these requirements. At the moment, three hand-crafted prototypes are available for experiments. At their core are three semiconductor accelerometers, one each for the X, Y and Z axis, that respond to changes in velocity producing a proportional voltage. The sensors in the MOVY prototypes can capture acceleration in the range of 4 mg to 2 g, up to 50 hz.


The MOVY prototype taken apart.

The analog signals are converted in a microprocessor that outputs, in a stream of serial (RS232)-frames, the raw acceleration data to a small radio transmitter. The corresponding receiver feeds the data into the serial port of a PC that runs the driver program which computes MOVY's orientation and location in X, Y and Z.

The first MOVY prototype including the radio transmitter draws about 40 milliamperes, ie, it can operate on a small battery. This allowed to fit the MOVY in a 45 x 50 x 38 mm case. To show the potential for further miniaturization, the latest MOVY is a modular, two accelerometer design: the sensors are fitted to a ring for the index finger, with a light cable running across the hand to the microprocessor and radio transmitter in a small box strapped to the wrist.

Performance

MOVYs are about one tenth the size of the trackers in use today. If mass-produced, MOVYs could be fairly cheap. MOVYs are good at determining orientation. Pitch and roll data have an accuracy of 1.4 degrees. The MOVY prototypes are less good at determining location: location data produced by integrating acceleration data are reliable only in specific situations. ■

Please contact:

Peter Henne – GMD
Tel: +49 2241 14 2688
E-mail: peter.henne@gmd.de

IQTension – Revealing Uneven Tension Profiles of Paper Reels

by Hannu Linna

In June 1998 Valmet Automation launched its IQTension product at the PulPaper '98 fair in Helsinki. IQTension provides a new method of revealing uneven tension profiles in cross direction (CD-profiles) of paper reels before they can cause any harm at the printing press. The system was developed by Valmet's various units in close collaboration with VTT Information Technology. Valmet is one of the world's leading supplier of paper and board machinery and related process control. So far Valmet Automation has sold nine IQTension systems, mostly to paper mills outside Finland. The first device was installed at Helprint Quebecor Oy, the largest rotogravure printing house in the Nordic countries.

According to VTT Information Technology's research results, the tension profile of the paper reel is an important quality property. There is a clear interdependency between the tension profile and the runnability of paper and thus also the amount of paper wasted at the printing press. The difference in the amounts of wasted paper is enormous when comparing good and bad paper reels. A large rotogravure printing house can lose tens or even hundreds of tons to wasted paper because of poor tension profiles meaning great economical loss.

During the development of Valmet's IQTension, the tension profiles of over 5,000 paper reels were measured at the Helprint Quebecor rotogravure printing house. The experiment revealed that the tension profile affected both the reel's break sensitivity and the amount of paper wasted during the printing process. The tension profile's minimum value measured at the beginning of the reel was the most critical factor for the number of breaks. During a run, the number of breaks depended considerably on how unbalanced the overall shape of the tension profile was. There are significant variations in the papers supplied by different manufacturers. The larger the fluctuations in the tension profile, the higher are the percentages of breaks. Harri Sundell, Production Manager at Helprint Quebecor Oy commented: "This product marks the beginning of a new era. Papermakers and printers can collaborate and use the information supplied by IQTension to achieve better quality in printing and less paper wastage".

IQTension is an unique on-line system for measuring web tension contactless and continuously across the entire width.

VTT has modelled the interdependencies between the different paper properties as well as the interactions between paper characteristics and runnability. The software, which makes it possible to take into consideration also properties measured across the web (CD-profiles), is currently customised for research purposes. It will, however, be developed into a practical tool with applications in every-day production. VTT Information Technology is a pioneer on a global scale as far as advanced research into the runnability of paper on paper machines and printing presses is concerned. VTT is also studying how greater advantage can be taken of the enormous amount of data measured from paper.

The first IQTension device installed in a paper mill is at UPM-Kymmene Corporation's Kaipola mill, where it has successfully improved both production efficiency and paper quality. The installation is also being exploited for research purposes.

IQTension was recently awarded the ATIP (Association Technique de l'Industrie Papetière) innovation prize at the ATIP 51st Annual Meeting in Grenoble, France. The ATIP innovation prize has been awarded four times. This year more than 30 products were competing for the prize. ■

Please contact:

Hannu Linna – VTT Information Technology
Tel: +358 0 456 5253
E-mail: Hannu.Linna@vtt.fi

Development of Research Networking in Africa

by Abraham Gebrehiwot and Stefano Trumpy

It is in the interest of the whole world for the African continent to become a full player in the global information society. In order to achieve this, consistent efforts and support are still needed from the more technologically advanced nations. CNUCE-CNR (previously) and IAT-CNR (now) have been strongly involved in such activities.

The RINAF (Regional Informatics Network for Africa) project was launched by UNESCO's Intergovernmental Informatics Programme (IIP) in 1992. Mainly financed by the Italian Government, the project has aimed at supporting the interconnection of academic and research institutions within Africa and their connection to the international research community through the provision of computer equipment, basic network services and the organization of training activities for technicians and end-users. The CNUCE Institute of the Italian National Research Council (CNR) was nominated as technical support agency; the newly constituted Institute for Telematics Applications (IAT-CNR), created from a department of CNUCE, is now responsible for project activities.

The original UNESCO-planned structure had the mandate to establish five regional (north, south, east, west and centre) and ten national nodes. The regional nodes (sited in Algeria, Kenya, Senegal, Nigeria and Zambia) had the task of managing and coordinating the activities of their region, establishing both regional connectivity and connections to the world-wide network.

In a recent meeting of the RINAF co-ordinators, held during the CARI

Conference in Dakar, October 1998, the support unit established at IAT-CNR, Pisa, presented their views on the major problems to be solved in the African Continent if research networks are going to be maintained and improved after the termination of RINAF activities. These views are outlined below.

The Data Communication Infrastructure

Communication costs are by far the most expensive part of a research network. Other expenses are the acquisition of the necessary hardware devices, the software procedures for providing the end-users with telematic services, and the costs of running the services, providing assistance to users and carrying out training activities. With respect to more technologically advanced countries, the cost of manpower in the African continent is significantly lower, while the cost for the communication infrastructure is normally much higher.

During a practical session of the RINAF regional course for Eastern Africa at Moi University in Eldoret - Kenya , 24-28 February 1997, with participants from Ethiopia, Uganda, Tanzania and Kenya.

It is harder to run research networks in Africa than in Europe, since governmental authorities do not support them, neither do they provide adequate support for the telecommunication costs of the local universities and research institutions. A number of African research networks are thus going commercial in order to maintain the service infrastructure; this means they rely on commercial services, provided by local PTT's or local Internet Service Providers, for the transport of data.

The situation in Africa is very mobile with respect to the data communications infrastructure; a number of efforts are currently working on fibre optics cabling throughout the continent in order to connect, as a first step, the capital cities. It is expected that, in the short-medium term, the strong pressure directed towards making Africa part of the global Information Society and, in particular, the commercial interests involved will allow the national research networks to gain from more efficient and less expensive data transmission capacities. The reality in Africa is that, in many cases, international links only reach the capital cities and some main urban centres while the rural community is left isolated. For this reason, radio links and satellite connections are needed to overcome the lack of local communication infrastructures.

Africa must take advantage of the Internet data transport protocols. The advantages are:

- adoption of standards which are widespread, low cost, interoperable, scalable, etc.
- employment of low cost entry level solutions which are robust and supported by the main providers
- insertion in a global technological effort thus opening up commercial opportunities to the African ISPs.

Capacity Building

Increased capacity building is needed; universities and research institutes should be assisted to maintain services and to pay adequate salaries to the most qualified staff. Different models for capacity building should be considered; the most successful are those developed by the Internet SOCiety (ISOC) and by the TransEuropean Research and Education Networking Association (TERENA).

When organizing training courses, particular attention should be given to the selection of trainees. Ideal candidates are highly motivated with an adequate technical background, and with an established position within their network organization once they return.

RINAF has dedicated a considerable part of the funds allocated for the first project phase to the organization of regional network training courses (held in 5 different African sub-regions). Other training courses will be organized at a national level.

Content Development

So far, not much has been done in this respect in Africa. From now on, a programme aimed at encouraging and guiding the local content development to prepare to play a relevant role in the 'Information Society' is needed. The technological gap today does not only regard the telecommunications infrastructures but also the information produced and made available.

The RINAF experience suggests that, in the interests of Africa, content development should:

- promote knowledge and cultural Internet activities as opposed to commercial ones which are taking off without the need for incentives
- promote communication between African nationals and expatriates in order to favour technology transfer and create the conditions for the return of expatriates to their home countries
- support the set up of web services
- promote projects aimed at supporting the interchange of data and the set up of common knowledge bases between a number of African countries.

Funding Models

It is very difficult to define a suitable funding model for a continent like Africa. Nevertheless, some general principals can be laid down:

- each country needs an organization involving the key players responsible for running the national research network
- each country needs a realistic programme based on available internal funds and on financial contributions from external sources

- Africa should set up organizations to coordinate international affairs/services and should become an equal partner of similar European research networking organizations (TERENA, DANTE, RIPE, CCIRN etc.)
- last but not least, a coordinated approach with the international funding bodies is needed.

■

Please contact:

Abraham Gebrehiwot – IAT-CNR
Tel: +39 050 593 336
E-mail: abraham@uoii.pisa.ccr.it

Tacton Configurator – Sales Configuration Support

by Lars Bergman

Matching customer requirements with combinations of components from a configurable products line is the core of the Tacton Configurator. As many companies are moving from selling products to selling systems this is a very important area for improving sales efficiency and effectiveness. Tacton Configurator has been developed within SICS, Swedish Institute of Computer Science, in the period of 1992 to 1997. The first industrialized version was released 1997. In 1998 a spin-off company, Tacton Systems, was established. Today the new company employs 15 people as a result of a flying start and rapid expansion. Main customers are four business units of Ericsson where some of the applications are in broad use.

When selling a complex system, as for instance telephone switching systems, composition of the 'right' system as viewed from customer needs and pricing is often a matter of weeks traditionally. Required is a mass of product information and input from knowledgeable people. With an application based on Tacton Configurator the reported time for configuring and pricing is reduced to seconds. This in turn has impact on the vendor competitiveness in a sales situation as well as it gives considerably improved efficiency in the sales process.

Generic Software

Tacton Configurator is a generic off-the-shelf software. It is adapted by creating a product configuration model and by connecting to other relevant software. The model is built by using a component oriented modelling language to describe configurable and fixed components at all levels as well as the properties of the components. Also the conditions for configuration has to be described in the model, as well as the parameters of customer requirements which are used to drive the end-user dialogue.

Research Base

Tacton Configurator is the result of research and development in the ISL, Intelligent Systems Laboratory within SICS, where the configurator was called Obelics. Domains to be mentioned are logic programming and constraints programming. The powerful and efficient programming systems SICStus Prolog and Oz, have been the enabling tools for the Obelics development. The research group makes the core of the new spin-off company, Tacton Systems, with Klas Orsvärn as the managing director who defended his thesis on 'Knowledge Modelling with Libraries of Task Decomposition Methods' in 1996. The Tacton website can be found at http://www.tacton.com

■

Please contact:

Klas Orsvärn – Tacton Systems
Tel: +46 8 690 07 50
E-mail: info@tacton.com

CALL FOR PAPERS

Third European Conference on Research and Advanced Technology for Digital Libraries

Paris, 22-24 September 1999

After Pisa in 1997 and Heraklion in 1998, ECDL'99 will take place in Paris at the prestigious location of the Bibliothèque Nationale de France. It is the third of a series of European conferences on research and technology for digital libraries.

The conference organisers solicit papers, panels and tutorials on innovative research on digital libraries, including but not limited to:

- digital library models, frameworks, and systems, interoperability, scalability
- information retrieval, navigation, indexing, catalogues
- multimedia information management, digitization (image, graphic, video, sound)
- electronic authoring, publishing, multilinguality
- metadata, knowledge representation, agent technologies
- experiments in DL system development, business models for digital libraries (pricing, etc.)
- user interfaces, evaluation of these interfaces by users.

Important dates:

- deadline for submission: 1 April 1999
- notification of acceptance: 1 June 1999
- papers due:1 July 1999.

The conference is sponsored by ERCIM; some scholarships will be provided for young researchers attending the conference. Selected papers will be published in the International Journal on Digital Libraries. For more information see the conference website at: http://www-rocq.inria.fr/EuroDL99/

■

Please contact:

Serge Abiteboul – INRIA
Tel: +33 1 3963 5537
E-mail: Serge.Abiteboul@inria.fr

CALL FOR PARTICIPATION

Electronic Communities – a New Engine for Regional Development – Connect '99

Trondheim, Norway, 18-20 May 1999

Connect 96, the first global summit on building electronic communities, took place at Stanford University in 1996. Europe was chosen to be the venue for the next conference in this series. In 1999 Trondheim, hometown of the Norwegian University for Science and Technology, will be the venue for Connect 99. This will continue the exchange of experience and visions for the future among pioneering leaders from industry, governmental bodies, research institutions and the political arena. The next conference is scheduled to take place in Japan.

Globalisation and the prevalence of information technology are predominant forces that enable economic decentralisation. The main question at Connect'99 will be how to utilise the technology to drive regional economic development. In particular, the focus will be on electronic commerce, learning opportunities and the role of changing organisations. Modern information networks provide new ways of conducting business, and information and knowledge will be available independent of geographical location. This will have a great impact on regional development.

The conference is sponsored by the Norwegian government, and the opening address will be presented by the Minister for Trade and Industry. The conference programme deals with major issues included in the Information Society Technologies (IST) Programme under the 5th EU Framework Programme, especially those covered by Key Action II. The conference will therefore serve as an important meeting place for potential participants in the IST Programme. Conference web site: http://www.sintef.no/connect99/ ■

Please contact:

Conference Secretariat – SINTEF
Tel: +47 73 59 26 45
Fax: +47 73 59 14 12
E-mail: connect99@sintef.no

CALL FOR PARTICIPATION

FMOODS'99 – Third International Conference on Formal Methods for Open Object-Based Distributed Systems

Florence, Italy, 15-18 February 1999

The goal of the FMOODS conferences is to provide a forum for the presentation of activities in three important and converging fields: formal methods, distributed systems and object-based technology.

The convergence of these areas is evidenced by recent advances in the field of distributed systems (for example, the ISO-ODP reference model for Open Distributed Processing and the work of the Object Management Group) and is creating links between several scientific communities. FMOODS'99 is the third in the series, initiated in Paris in March 1996 and continued in Canterbury, July 1997. The conference is supported by IFIP, in particular TC 6/WG6.1, and sponsored by CNR, EU-DGXIII, and the Universities of Florence and Bologna. It will be held in a beautiful site in the hills overlooking Florence, an easy ride from the city centre.

The conference programme includes 3 tutorials and 5 invited talks by international experts on the hot issues of the fields covered by the conference, as well as 26 refereed papers. For the programme and for detailed registration and accomodation instructions see the Conference web site: http://www.dsi.unifi.it/fmoods/ ■

Please contact:

Paolo Ciancarini and Roberto Gorrieri, Programme Co-Chairs – University of Bologna
Tel: +39 050 354 509
Fax: +39 050 354 510
E-mail: {ciancarini,gorrieri}@cs.unibo.it

Alessandro Fantechi, Organizing Chair – University of Florence
Tel: +39 05 5479 6265
Fax: +39 05 5479 6363
E-mail: fantechi@dsi.unifi.it

CALL FOR PAPERS

IWOSS'99 – International Workshop on Similarity Search

Florence, Italy, 1-2 September 1999

The Workshop seeks contributions elaborating on all aspects of similarity search, ranging from theoretical work (including computational geometry and learning theory) to practical experiences. Position papers suggesting new research directions or specifying new application needs are of particular interest. Specific topics of interest include:

- similarity search models and paradigms
- theoretical aspects and properties of similarity retrieval and indexing
- complex similarity queries and query optimization
- similarity search indexes
- performance studies, benchmarking
- approximate similarity retrieval
- similarity search and browsing
- relevance feedback for similarity retrieval
- similarity search with respect to the quality of service
- new applications of similarity retrieval
- practical experiences, etc.

Important Dates:

- abstract submission deadline: 23 March 1999
- paper submission deadline: 30 March 1999
- notification: 25 May 1999
- camera-ready copy due: 15 June 1999.

For more information, see the Conference Web Site: http://www-db.deis.unibo.it/IWOSS99 ■

Please contact:

Fausto Rabitti – CNUCE-CNR
Tel: +30 050 593 396
Fax: +39 050 904 052
E-mail: F.Rabitti@cnuce.cnr.it

CALL FOR PAPERS

Fourth ERCIM Workshop on Formal Methods for Industrial Critical Systems

Trento, Italy, 11-12 July 1999

The Fourth ERCIM Workshop on Formal Methods for Industrial Critical Systems will be held in Trento on July 11-12, 1999, as a satellite meeting of FLoC'99 - Federated Logic Conference (see http://www.cs.bell-labs.com/cm/cs/what/floc99/). The aim of the FMICS workshops is to provide a forum mainly for, but not limited to, researchers of ERCIM sites who are interested in the development and application of formal methods in industry. In particular, these workshops should bring together scientists that are active in the area of formal methods and interested in exchanging their experiences in the industrial usage of these methods. They also aim at the promotion of research and development for the improvement of formal methods and tools for industrial applications.

Papers must be in English, no more than 25 pages long,including abstract and keywords. Electronic submission by email to fmics@iei.pi.cnr.it is encouraged provided that the format is encapsulated standard Postscript printable by any postscript device.

Important Dates

Deadline for submission: 1 March 1999
Notification of acceptance: 30 April 1999
Final manuscript: 30 May 1999

For further information, see http://www.cnuce.pi.cnr.it/cnuweb/research/resgroups/conc-meth/FMICS/ ■

Please contact:

Stefania Gnesi – IEI-CNR
Tel: +39 050 59 34 89
E-mail: gnesi@iei.pi.cnr.it

CWI –**Hans van Duijn**, leader of CWI's research cluster on Modelling, Analysis and Simulation, and professor in the Mathematical Analysis of Flows through Porous Media at Delft University of Technology, received the Max Planck Research Award for International Cooperation. The award of DM 250.000 was presented during a festive ceremony on 3 December in Bonn. It was granted in recognition of Van Duijn's outstanding research achievements and provided for cooperation with German scientists over a period of three to five years.

CWI – **a CWI research team led by Dick Bulterman** has won one of the three main prizes (Hfl 50.000) of the

From left to right: Dick Bulterman, Jack Jansen, Lynda Hardman (all CWI), and Morris Tabaksblat, Chairman and CEO Unilever N.V.

McKinsey New Venture 98 competition in The Netherlands for the best business plan of prospective entrepreneurs in the research world. The team (Dick Bulterman, Lynda Hardman, Sjoerd Mullender, and Jack Jansen) won the prize with their idea of GRiNS, an authoring system for multimedia applications on the Internet (see, eg, ERCIM News 33, p.45). GRiNS can be used to develop the full potential of SMIL (Synchronized Multimedia Integration Language), which became a W3C Recommendation in June 1998. There were over six hundred submissions for the competition, which was organized in three rounds by McKinsey in cooperation with the Dutch universities, research institutions and the Twinning Network.

CWI – **Ronald Cramer** received the Christiaan Huygens Award (Hfl 20.000), granted by the Royal Netherlands Academy of Arts and Sciences for the most innovative Ph.D. thesis over the past four years in The Netherlands in the field of Information & Communication Technology. Ronald Cramer worked at CWI under Paul Vitányi. After having

Photo: Benelux Press.

Ronald Cramer (left) receiving the Christiaan Huygens Prize.

completed his Ph.D. thesis on secure, yet practical cryptographic systems he went early 1997 to ETH Zürich to continue research in this direction. Jointly with Victor Shoup (IBM Research Laboratory Zürich) he devised a 'hacker-proof' encryption system which protects, eg, Internet transactions against so-called active attacks (successful attacks on the best encryption systems so far were carried out earlier in 1998). The result was announced at the Crypto'98 conference held in September in Santa Barbara, California and received wide publicity.

CWI – **Debby Lanser**, now a Ph.D. student in CWI's research theme on Numerical Algorithms for Air Quality Modelling (led by Jan Verwer), received at Delft University of Technology the annual prize (Hfl 1.000) for the best Master's Thesis in the field of Technical Mathematics. The work concerned the modelling and computing of a complex industrial flow problem.

INRIA – **Olivier Faugeras**, Research Director at INRIA and head of the ROBOTVIS research team at Sophia-Antipolis adressing the computer-aided vision issue, has been elected as full member of the French Academy of Science. A former graduate of Ecole Polytechnique, he holds a PhD from Utah University and from University Paris 6. He is a part-time professor at MIT and lectures in various French universities. His book 'Three-dimensional Computer Vision:a Geometric Approach' has become a classic.

European Research Consortium for Informatics and Mathematics

ERCIM

The European Research Consortium for Informatics and Mathematics (ERCIM) is an organisation dedicated to the advancement of European research and development, in the areas of information technology and applied mathematics. Through the definition of common scientific goals and strategies, its national member institutions aim to foster collaborative work within the European research community and to increase co-operation with European industry. To further these objectives, ERCIM organises joint technical Workshops and Advanced Courses, sponsors a Fellowship Programme for talented young researchers, undertakes joint strategic projects, and publishes workshop, research and strategic reports, as well as a newsletter.

ERCIM News is the in-house magazine of ERCIM. Published quarterly, the newsletter reports on joint actions of the ERCIM partners, and aims to reflect the contribution made by ERCIM to the European Community in Information Technology. Through short articles and news items, it provides a forum for the exchange of information between the institutes and also with the wider scientific community. ERCIM News has a circulation of 7,000 copies.

ERCIM Central Office:
Domaine de Voluceau
Rocquencourt
B.P. 105
F-78153 Le Chesnay Cedex
FRANCE
E-mail: office@ercim.org

Central Editor:	E-mail:	Telephone:
Peter Kunz	peter.kunz@ercim.org	+33 1 3963 5040
Local Editors:		
Grabriela Andrejkova (SRCIM)	andrejk@kosice.upjs.sk	+421 95 6221128
Lars Bergman (SISU)	lars@sisu.se	+46 8 752 1613
Erzsébet Csuhaj-Varjú (SZTAKI)	csuhaj@sztaki.hu	+36 1 209 6990
Truls Gjestland (SINTEF)	truls.gjestland@informatics.sintef.no	+47 73 59 26 45
Michal Haindl (CRCIM)	haindl@utia.cas.cz	+420 2 6605 2350
Fritz Henglein (DANIT)	henglein@diku.dk	+45 35 32 14 02
Bernard Hidoine (INRIA)	bernard.hidoine@inria.fr	+33 1 3963 5484
Pia-Maria Linden-Linna (VTT)	pia-maria.linden-linna@vtt.fi	+358 0 456 4501
Siegfried Münch (GMD)	siegfried.muench@gmd.de	+49 2241 14 2255
Henk Nieland (CWI)	henkn@cwi.nl	+31 20 592 4092
Carol Peters (CNR)	carol@iei.pi.cnr.it	+39 050 593 429
Martin Prime (CLRC)	martin@inf.rl.ac.uk	+44 1235 44 6555
Constantine Stephanidis (FORTH)	cs@csi.forth.gr	+30 81 39 17 41

You can subscribe to ERCIM News free of charge by: sending e-mail to your local editor; posting paper mail to the address above or filling out the form at the ERCIM web site at http://www.ercim.org/

CLRC

Central Laboratory of the Research Councils

Rutherford Appleton Laboratory
Chilton, Didcot
GB-Oxon OX11 0QX

Tel: +44 123582 1900
Fax: +44 1235 44 5385
http://www.cclrc.ac.uk/

CWI

Centrum voor Wiskunde en Informatica

Kruislaan 413
NL-1098 SJ
Amsterdam

Tel: +31 205929333
Fax: +31 20 592 4199
http://www.cwi.nl/

CNR

Consiglio Nazionale delle Ricerche

IEI-CNR
Via S. Maria, 46
I-56126 Pisa

Tel: +39 050 593 433
Fax: +39 050 554 342
http://www.iei.pi.cnr.it/

Crcim

Czech Research Consortium for Informatics and Mathematics

FI MU
Botanicka 68a
CZ-602 00 Brno

Tel: +420 2 6884669
Fax: +420 2 6884903
http://www.utia.cas.cz/CRCIM/home.html

DANIT

Danish Consortium for Information Technology

DANIT co/CIT
Aabogade 34
DK - 8200 Aarhus N

Tel: +45 8942 2440
Fax: +45 8942 2443
http://www.cit.dk/ERCIM/

FORTH

Foundation for Research and Technology – Hellas

Institute of Computer Science
P.O. Box 1385
GR-71110 Heraklion, Crete

Tel: +30 81 39 16 00
Fax: +30 81 39 16 01
http://www.ics.forth.gr/

GMD

GMD – Forschungszentrum Informationstechnik GmbH

Schloß Birlinghoven
D-53754 Sankt Augustin

Tel: +49 2241 14 0
Fax: +49 2241 14 2889
http://www.gmd.de/

INRIA

Institut National de Recherche en Informatique et en Automatique

B.P. 105
F-78153 Le Chesnay

Tel: +33 1 39 63 5511
Fax: +33 1 39 63 5330
http://www.inria.fr/

SICS

Swedish Institute of Computer Science

Box 1263
S-164 28 Kista

Tel: +46 8 633 1500
Fax: +46 8 633 7230
http://www.sics.se/

SARIT

Swiss Association for Research in Information Technology

Dept. Informatik
ETH-Zentrum
CH-8092 Zürich

Tel: +41 1 632 72 41
Fax: +41 1 632 11 72
http://www-dbs.inf.ethz.ch/sarit/

SINTEF Telecom and Informatics

Stiftelsen for Industriell og Teknisk Forskning ved Norges Tekniske Høgskole

SINTEF Telecom & Informatics
N-7034 Trondheim

Tel :+47 73 59 30 00
Fax :+47 73 59 43 02
http://www.informatics.sintef.no/

SRCIM

Slovak Research Consortium for Informatics and Mathematics

Dept.of Computer Science, Comenius University
Mlynska Dolina M
SK-84215 Bratislava

Tel: +421 7 726635
Fax: +421 7 727041

Magyar Tudományos Akadémia – Számítástechnikai és Automatizálási Kutató Intézete

P.O. Box 63
H-1518 Budapest

Tel: +36 1 4665644
Fax: + 36 1 466 7503
http://www.sztaki.hu/

VTT

Technical Research Centre of Finland

VTT Information Technology
P.O. Box 1200
FIN-02044 VTT

Tel:+358 9 456 6041
Fax :+358 9 456 6027
http://www.vtt.fi/

Directeur de Publication: B. Larrouturou
Publication périodique réalisée par GEIE-ERCIM

ISSN 0926-4981