Number 88, January 2012

ERCIM NEWS

www.ercim.eu

Special theme:

Evolving Software

Also in this issue:

ERCIM News is the magazine of ERCIM. Published quarterly, it reports on joint actions of the ERCIM partners, and aims to reflect the contribution made by ERCIM to the European Community in Information Technology and Applied Mathematics. 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. This issue has a circulation of about 8,500 copies. The printed version of ERCIM News has a production cost of €8 per copy. Subscription is currently available free of charge.

ERCIM News is published by ERCIM EEIG
BP 93, F-06902 Sophia Antipolis Cedex, France
Tel: +33 4 9238 5010, E-mail: contact@ercim.eu
Director: Jérôme Chailloux
ISSN 0926-4981

Contributions
Contributions must be submitted to the local editor of your country

Advertising
For current advertising rates and conditions, see http://ercim-news.ercim.eu/ or contact peter.kunz@ercim.eu

ERCIM News online edition
The online edition is published at http://ercim-news.ercim.eu/

Subscription
Subscribe to ERCIM News by sending email to en-subscriptions@ercim.eu or by filling out the form at the ERCIM News website: http://ercim-news.ercim.eu/

Next issue
April 2012, Special theme: "Big Data"

Change is the Constant

There is something peculiar about software. When it idles, it does not rust. While in use, it does not wear down. Nonetheless, from the moment a software system is released, its quality deteriorates and an incessant stream of fixes, patches, and enhancements is required to keep its users satisfied. What's going on?

Software systems stop meeting the needs they were built to satisfy, not because the software changes, but because the needs change. Let's review some of the external drivers for change:

- **Innovation:** Businesses compete by bringing new or improved products and services to the market. Software supports the production of products and the delivery of services. Business innovation drives software change.
- **Cost reduction:** Services and products that were once innovative lose their differentiating power when competitors start offering the same for less. In markets where similar products or services compete on price, the operational costs of the software systems that support them become a critical factor. Reduction of operational costs drives software change.
- **Regulation:** Governments are constantly at work to change laws and regulations, be it for the betterment of society or for propping up the financial system. Such changes in the rules require changes not only in the governmental software systems that enforce them, but also in the software systems of banks, airlines, and other businesses that must comply with these rules. Laws and regulations drive software change.

Such external factors drive software change directly. To make things worse, they also cause change through indirect mechanisms.

The internal complexity of software systems is overwhelming. The program code of a medium-sized software system may easily fill 3000 pages of A4 paper, but does not form a linear story. Instead, all paragraphs are intricately interlinked by cross-references. When making a change to such a complex structure, bugs are inevitably introduced. Thus, the initial change indirectly leads to the need for further changes down the line.

Each software system is dependent on others. For example, a web store depends on a payment system, a database system, an internet browser, several operating systems, and so on. Changes in any of these systems may induce the need for changes in the system that depends on them. Thus, changes in one system propagate to other systems through the network of dependencies among them.

As the pace of change of society and business increases, the need for software change increases. And as software systems become more complex and more interdependent, changing them becomes harder. This is the squeeze that we are in and that software evolution research seeks to alleviate.

Joost Visser, Head of Research, Software Improvement Group

Several fundamental challenges need to be met:

- **Knowledge dissipation:** You cannot change a system that you do not understand. To enable evolution, software developers need to be supported in acquiring an understanding of the code they need to change.
- **Entropy:** Changing software in an uncontrolled manner will make it more complex. More complex software is harder to change in a controlled manner. To break this vicious cycle, mechanisms are needed to control complexity during change.
- **Synchronization:** The interdependencies between software systems require that changes be synchronized. Software evolution must be supported with mechanisms that loosen the coupling between systems or automate the propagation of changes from one system to another.

As you will read in this issue, researchers are hard at work to confront these challenges.

As solutions become available, in the form of tools and methods for evolving software, a new challenge arises: How can organizations best apply these solutions to manage the evolution of their software portfolios?

To answer this question, a deeper understanding is needed of the economics of software evolution. Changing a software system, or replacing it, represents an investment of time and resources with potential returns in terms of innovation, cost reduction, or regulatory compliance. The application of new solutions for evolving software will impact the cost, risk, speed, and even the outcome of evolution.

These economic factors must be properly understood, quantified, and combined to support the decision-making process regarding software evolution at the software portfolio level. And one economic principle of software evolution must be understood by all: Software is not built to last; it is built to change.

Joost Visser

RESEARCH AND INNOVATION

This section features news about research activities and innovative developments from European research institutes

EVENTS, BOOKS, IN BRIEF

Engaging IT and Fishery Specialists for Enhanced Marine Living Resource Conservation

by Hilary Hanahoe

The iMarine project is chartered with establishing an open data infrastructure that will effectively support the implementation of the ecosystem-based approach (EA) to fisheries management and conservation of marine living resources.

iMarine's ultimate goal lies in defining corrective actions, alleviating pressure from endangered ecosystems, and assisting policy makers in promoting the sustainable exploitation of marine ecosystems. In the words of Donatella Castelli, ISTI-CNR, iMarine project director, "the iMarine data infrastructure will foster innovation by providing an open platform and a variety of services that are designed to become an integral part of the organized procedures of a wide community of practitioners addressing the challenges of fisheries management and the conservation of our marine living resources".

iMarine's roots lie in the D4Science project, which has developed capacities for data interoperability among different research infrastructures. The project has delivered an open-source technology capable of operating data e-Infrastructures in a federated virtual environment. D4Science identified the sharing of data among different partners as the next challenge. This is the engine behind iMarine, as Marc Taconet, FAO & iMarine Board Chair, explains "the idea was to give stronger capacity to drive the directions of data infrastructures development to an identified Community of Practice".

iMarine's Community of Practice, which plays a central role in underpinning the project's goal, brings on board both people working in biodiversity and people dealing with fisheries management, such as scientists, managers, lawyers, industry leaders, unions, fishery operators, sharing the same goal of fostering the sustainable supply of our living resources for future generations. The iMarine Board serves as an interface between the Community of Practice that supports the ecosystem approach and technology developers who ensure user requirements are taken on board. Almost 50% of fishery resources in our oceans are approaching the limits of sustainability. Another 28% are either overfished or hovering near depletion because of man-made and natural disasters. Good fishery management covering social, nutrition and economic facets becomes imperative to ensure our ocean's supply is sustainable now and in the future. This grand global challenge needs to be addressed by planning, developing, and managing fisheries in a way that caters for multiple social needs through a coordinated approach within ecologically meaningful boundaries. iMarine will establish and operate a data infrastructure that will facilitate the interdisciplinary collaboration among the members of the Community of Practice and will encourage the emergence of new scientific approaches and more effective working practices. Economies of scale afforded by the use of a common data infrastructure will significantly reduce the cost of applying the principles of the Ecosystem Approach. iMarine thus contributes to the three fundamental sustainability pillars – environmental, social and economic – of this Ecosystem Approach to Fisheries.

Implementing an ecosystem approach to fisheries management is intimately bound up with data sources and knowledge generation which is far broader in scope than traditional fisheries management and conservation. It is imperative that monitoring and assessment of target, emblematic, or vulnerable species be broadened to cover species assemblages, communities, habitats, and ecosystems, and that socio-economics be also broadened to cover fisheries' impacts on all goods and services offered by those ecosystems. What's more knowledge sources are generated in a wide variety of formats and stored in a large number of repositories, archives, and databases reflecting different policies, practices, and standards, which will all be addressed by iMarine. Providing a common layer to connect existing data sources and data infrastructures via established and standardized interfaces, implementing new features and value-added tools used by practitioners in fisheries management and marine living resources conservation allows the creation of a unique reference point for the global EA community of practitioners.

iMarine kick-off meeting participants

iMarine sets sail

The iMarine project was officially launched on 16 November 2011 in Pisa, where over 40 representatives from both the consortium and the iMarine board met for the first time. The significance of this first meeting is summed up by Marc Taconet, "this is the first time that we brought together such a rich set of high level experts with different expertise. The discussion was very rich and continued in a fruitful brain-storming that we are going to organise in collaborative remote discussions from now on. Clear directions have already been identified in terms of interoperability standards. We are confident that this board will help reach our goals".

The iMarine project is co-coordinated by the ERCIM Office and CNR-ISTI.

Link: http://www.i-marine.eu

Please contact:
Donatella Castelli, ISTI-CNR, Italy, iMarine project director
E-mail: donatella.castelli@isti.cnr.it

Jessica Michel, ERCIM, France
iMarine administrative and financial director
E-mail: jessica.michel@ercim.eu

ERCIM Alain Bensoussan Fellowship Programme

The ERCIM Alain Bensoussan Fellowship Programme enters the fourth round of fellowships supported by the COFUND programme of the European Commission with the next application deadline 30 April. Some 90 fellowships have already been granted under the COFUND scheme.

Who can apply?

The fellowships are available for PhD holders from all over the world.

What is the duration?

Fellowships are either of 24 months duration spent in two of the ERCIM institutes, or of 12 months duration spent in one institute.

Application deadlines:

Twice per year:
30 April and 30 September.

How to apply?

Only online applications are accepted.
The application form will be online one month prior to the application deadline.

Which topics/disciplines?

Topics cover most disciplines in computer science, information technology, and applied mathematics.

Where are the fellows hosted?

Fellows can be hosted at ERCIM member institutes only (the current ERCIM member institutes are listed on the back page of this issue). When an ERCIM member is a consortium (AARIT, CRCIM, PEG, PLERCIM, SARIT, SpaRCIM), the hosting institute might be any of the consortium's members. When an ERCIM Member is a funding organisation (FNR, FWO/FNRS), the hosting institute might be any of their affiliates.

What are the conditions?

- have obtained a PhD degree during the last eight years (prior to the application deadline) or be in the last year of the thesis work
- be fluent in English
- be discharged or get deferment from military service
- the fellowship is restricted to two terms (one reselection possible)
- have completed the PhD before starting the grant.
- a member institute cannot host a candidate of the same nationality
- a candidate cannot be hosted by a member institute, if by the start of the fellowship, he or she has already worked in this institute for a total of six months or more, during the last three years.

How are the fellows selected?

Each application is reviewed by scientists, and the criteria for selection are:

- scientific expertise of the applicant
- quality of scientific publications
- relevance of the fellow's research agenda
- interest/added-value for the ERCIM consortium
- previous mobility / professional experiences.

The number of available positions depends on the needs of the member institutes and their available funding.

More information: http://fellowship.ercim.eu/

Introduction to the Special Theme

Evolving Software

by Tom Mens and Jacques Klein

As chair and member of the ERCIM Working Group on Software Evolution, we are honoured to act as guest editors for this issue of ERCIM News with the special theme of Evolving Software. The importance of software in our information society cannot be underestimated: just imagine what would happen if all software systems around us failed. There would be no television, no radio, no Internet, no financial transactions and e-commerce, no energy production, huge problems in public transport, and the list goes on.

In the late 1960s it became clear that in order to cope with the rising complexity of software systems, more disciplined techniques were needed. This need gave rise to the first international conference on Software Engineering, organized by the NATO Science Committee, in 1968. Its goal was "the establishment and use of sound engineering principles in order to obtain reliable, efficient and economically viable software". Inspired by this, Belady and Lehman explored and emphasized the role of program evolution and evolution dynamics in the seventies. Their work gave rise to Lehman's "laws of software evolution", an area that is still under active study today by many researchers. The need for software evolution was also acknowledged by other authorities, such as Fred Brooks who stated in his book "The Mythical Man-Month" that any successful software will inevitably need to be maintained in order to remain viable over time.

Today, four decades later, the relevance of software evolution has only increased. Software evolution will always remain inevitable due to a wide variety of factors: users that wish to have new features, bugs that need to be fixed, market pressure by competitors, technological changes in the environment with which the software needs to interact, performance issues that need to be dealt with, and so on. To accommodate these change requests, the software product needs to be changed and improved on a regular basis. In parallel to this, the software process itself is also subject to improvement, in order to produce new versions of the software more quickly and cost-effectively, without compromising software quality.

As can be witnessed from the keynote by Joost Visser from SIG earlier in this issue, the need to evolve software is of utmost importance to all software-producing and software-consuming companies. The aim of this theme issue of ERCIM News is therefore to raise awareness of this need, and to present a portfolio of scientific research on evolving software that is currently ongoing in many research groups all over Europe. Although only a few of the many European research groups actively involved in software evolution have contributed to this theme issue, the presented articles provide a representative and well-balanced overview of what is going on in the field. They cover a wide range of activities:

- "Understanding" how the software has evolved in the past; how development teams collaborate to evolve software more effectively; and so on
- "Modelling" the process of software evolution
- "Predicting" how the software product quality (eg in terms of defects) will evolve in the future

- "Controlling" the evolution process (eg Do we deliver new releases in time and within budget? Do the releases fix bugs and include new features requested by the user?) and the product (eg Is the product of sufficient quality and performance?)
- "Automating" the evolution process (eg by using workflow software, by detecting and reporting process deviations on the fly), by providing real-time recommendations to developers (such as code completion, conflict detection)
- "Visualizing" software evolution
- "Improving" the software process and the software product

This theme issue includes four exciting invited papers written by recognized European researchers in the field: Michele Lanza (Switzerland), Xavier Blanc (France), Alexander Serebrenik (The Netherlands) and Theo D'Hondt (Belgium). These papers go from a holistic view of software evolution (dealing with a software system as a whole to better see the forest for the trees), to more specific application domain such as software for the Internet and the use of new measurement techniques for software quality.

Twenty-six regular papers have been selected. They form a representative palette of research projects in the field at both European and national levels. They also demonstrate the importance of software evolution through concrete case studies, and show that software evolution does not focus only on pure IT systems (object-oriented and service-oriented software, software-intensive networked systems, legacy systems, free and open source software, middleware etc) but is also extremely relevant in application domains such as automotive, molecular modelling, information systems, geographic information systems, mobile computing, and so on.

This selection of papers gives a clear overview of both traditional and emerging software engineering techniques, tools and approaches used by software evolution experts. Examples of such techniques are software/data analysis, data visualization, software metrics, software architecting, metaprogramming, and more general paradigms like model-driven engineering (MDE) and software product line engineering (SPLE). These techniques provide a portfolio for tool developers to build dedicated software evolution tools. These tools are vital in order to better understand, manage and anticipate software evolution.

We hope that these articles will convince you of the importance, relevance and challenges in software evolution research and practice in Europe. Enjoy!

Link:
ERCIM Software Evolution Working Group:
http:/wiki.ercim.eu/wg/SoftwareEvolution

Please contact:
Tom Mens,
University of Mons, Belgium
E-mail: tom.mens@umons.ac.be

Jacques Klein
University of Luxembourg
E-mail: jacques.klein@uni.lu

Further Reading:
Software Evolution, T. Mens and S. Demeyer (Editors), Springer, 2008. ISBN 978-3-540-76439-7.

Software Evolution: Maintaining Stakeholders' Satisfaction in a Changing World. T. Mens, Y.-G. Guéhéneuc, J. Fernández-Ramil, M. D'Hondt (Guest Editors), IEEE Software 27: 4. 22-25 July/August, 2010

Upcoming conferences:
CSMR 2012: 16th IEEE European Conference on Software Maintenance and Reengineering, Szeged, Hungary, 27-30 March 2012

ICSM 2012: 28th IEEE International Conference on Software Maintenance, Trento, Italy, 23-30 September 2012

Holistic Software Evolution

by Michele Lanza

In order to analyze and understand large, complex, and evolving software systems, we take a holistic stance by combining diverse sources of information.

Software evolution research has two objectives: given a system's present state, evolutionary information is used to understand its past, and predict its future.

But, where does evolutionary information come from? Traditionally, developers use software configuration management (SCM) systems, such as SVN and git. These systems record snapshots of the code, and store them in code repositories, which represent access and synchronization points for development teams. Given the widespread adoption and public availability of SCM systems, it is no wonder that researchers have recently been focusing on mining such software repositories, creating a new and challenging research field.

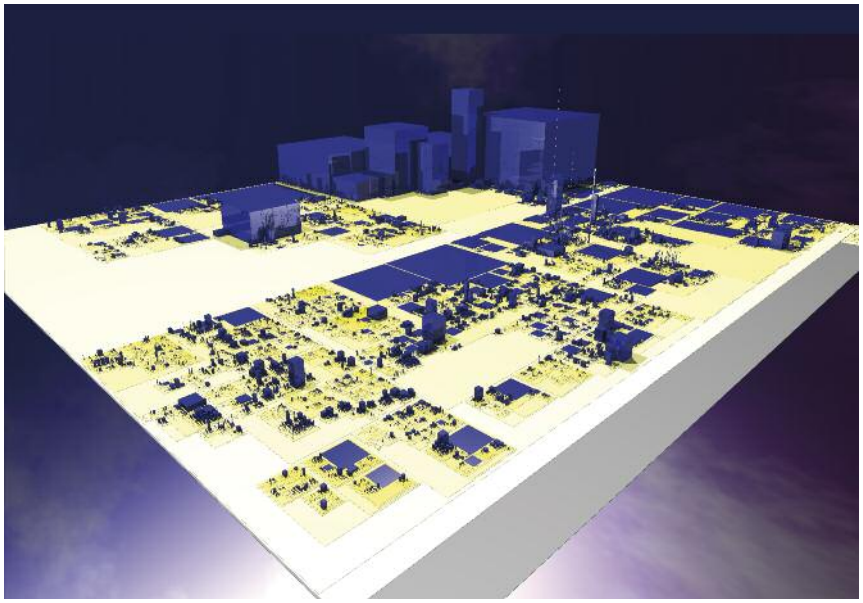


Figure 1: A depiction of the Eclipse IDE (approx. 4 million lines of Java source code) as a software city. The 1,900 packages making up the system are rendered as districts, which contain close to 29,000 classes, rendered as buildings. The height of the buildings represents the number of methods of the classes; the base size represents the number of variables.

With software evolution data it is easy not to see the wood for the trees. Besides SCM systems, there is a great deal of highly relevant data recorded in different types of repositories, such as bug trackers, mailing lists, newsgroups, forums, where system-relevant information is stored. The plethora and diversity of data represent a major research challenge in terms of how to extract relevant information and how to present it in a meaningful and effective fashion.

REVEAL - The REVEAL group at the University of Lugano approaches research from a holistic angle: The central theme is to piece together the various sources of information in a comprehensive whole, to have a better grip on the complex phenomenon known as software evolution. We are striking three new research paths to unify, reify, and see software evolution.

Unify - It is not obvious how to integrate information that comes from various sources and in a variety of forms. While certain repositories are easy to mine (especially source code, where the data is well structured), others store "humane" information, written by humans for humans, such as emails. We are currently experimenting with techniques dealing with fault-tolerant parsing (such as island parsing) and information retrieval (IR) to model the data in a unified way.

The PhD thesis of Marco D'Ambros (2010) demonstrated that by integrating various types of data, such as evolutionary data, code-specific data, and data on defects, it is possible to develop efficient defect prediction techniques that go beyond the state of the art. The ongoing work of Alberto Bacchelli is providing evidence for the usefulness of humane data: understanding what developers communicate about is often crucial for the understanding of what they produce.

Reify - Researchers often face the problem that the data produced by current mainstream versioning systems is of poor quality. The snapshot-based nature of SCM systems induces information loss: evolutionary data is only recorded when developers explicitly commit their changes to the SCM system. What happens in between is lost forever. Allegorically speaking this is similar to watching a movie when only every 10th frame is shown. While it may still be possible to understand the general theme of the movie, it certainly does not make for a great viewing experience.

We investigated this issue through the PhD theses of Romain Robbes (2008) and Lile Hattori (2012). The common theme is to complement classical versioning with information constantly recorded while developers work in their integrated development environment (IDE). The central idea is to embrace change by reifying it into a first-class concept. The consequence is almost perfect historical data. Through a number of studies and experiments we have demonstrated that the crystalline nature of the obtained information can fuel real-time recommender systems (such as code completion, conflict detection, etc.), which proactively help developers in their daily chores. Our

vision is to turn the "I" in IDE into "Intelligent".

See - What should we do with holistic information, once it's there? The challenge resides in the sheer bulk of data, and what it all means to a developer who is hardly interested in having more information at hand. The point is to have useful, understandable and actionable data. We believe that software visualization will play a major role in this context. It is the graphical depiction of software artifacts, and leverages the most powerful sense humans have: vision. While many shy away from the human-centric nature of this type of research, it is based on surprisingly simple notions that leverage how our brain processes visual input. The PhD thesis of Richard Wettel (2010) successfully explored a city metaphor to depict complex and evolving software systems, demonstrating that it is indeed possible to visually grasp a phenomenon as complex as software evolution.

Beyond - Software evolution research is only slowly coming of age, turning into a research field whose intrinsic complexity is further augmented by the fast pace of technical innovations. A holistic take is the only chance to prepare for unexpected consequences.

Please contact:
Michele Lanza
University of Lugano, Switzerland
Tel: +41 58 666 4659
michele.lanza@usi.ch
http://www.inf.usi.ch/lanza/

A One-Stop-Shop for Software Evolution Tool Construction

by Mark Hills, Paul Klint, Tijs van der Storm and Jurgen Vinju

Real problems in software evolution render impossible a fixed, one-size-fits-all approach, and these problems are usually solved by gluing together various tools and languages. Such ad-hoc integration is cumbersome and costly. With the Rascal meta-programming language the Software Analysis and Transformation research group at CWI explores whether it is feasible to develop an approach that offers all necessary meta-programming and visualization techniques in a completely integrated language environment. We have applied Rascal with success in constructing domain specific languages and experimental refactoring and visualization tools.

The goal of our research is to rapidly construct many different (experimental) meta-programs that perform software analysis, transformation, generation or visualization. The underlying motivations are (a) to do experimental research in software evolution and construction and (b) to transfer our results to industry in the shape of tools. We need innovative tools that contribute to understanding and improving very complex software systems.

The challenge is diversity: the mixture of programming languages and data formats one encounters is overwhelming when solving problems in software evolution. The range of questions to answer is even larger; they range from general (eg computing a call graph or a widely used metric) to application specific (eg checking in-house coding standards or migrating to a different API). As a result, software evolution problems are often solved with different tools that are glued together. The gluing itself introduces yet another kind of complexity.

Rascal is a domain-specific language that tackles this challenge by integrating all aspects of meta-programming. It is a language for performing any or all meta-programming tasks such as the analysis, transformation and visualization of existing source code and models and the implementation of domain-specific languages. Rascal is completely written in Java, integrates with Eclipse and gives access to programmable IDE (Interactive Development Environment) features using IMP (the Eclipse IDE Meta-Tooling Platform). We are aiming to provide a complete "One-Stop-Shop" for analysis, transformation, generation and visualization of software systems.

Rascal has many applications in the software evolution domain, for example, when studying the effects of choosing between design patterns. Design patterns provide reusable, named solutions for problems that arise when designing object-oriented systems. While in some cases it is clear which pattern should be used, in others multiple patterns could apply. When this happens, the designer has to carefully weigh the pros and cons of each option as applied both to the current design and to plans for future evolution of the system.

The specific design choice we focused on was between structuring AST-based (Abstract Syntax Tree based) language interpreters according to either the Visitor or the Interpreter design pattern. While it seems clear that either pattern will suffice from a functional point of view, it is unclear what the non-functional quality of the interpreter will be in each case. In theory, the Interpreter pattern might have lower method call overhead because it does not involve double dispatch, it should allow easier extension with new language features, and it should be easier to add local state to AST nodes. In theory, the Visitor pattern should allow easier extension with new kinds of operations on AST nodes and should allow better encapsulation of state required by such operations. The question remains how this choice affects the design in practice, given the plethora of other factors of software maintainability and efficiency.

Our research method is as follows. Given is an implementation of an inter-

preter (in this case study the Rascal interpreter itself) that uses Visitor in several places. Using Rascal, we construct an automated source-to-source transformation that transforms the Java code of each instance of Visitor to an instance of the Interpreter design pattern. This then allows us to conduct a comparison between two implementations varying only in the choice of design pattern. The choice has been isolated among all the other factors in software design. We can now compare ease of maintenance and runtime performance between the two semantically equivalent systems.

This method enabled us to identify the differences between using the Visitor and Interpreter patterns in the interpreter. We applied real maintenance scenarios to both versions and did initial performance measurements. There was no observable difference in efficiency, but we did reach some counter-intuitive conclusions regarding maintainability: Visitor wins in the long run, even where Interpreter should be better. While these results cannot be directly generalized, we expect that other designers of tree-centric object-oriented software — such as compilers, interpreters, and XML processors — will benefit.

Rascal was used to rapidly (two person-weeks) create this complex ad-hoc Java source-to-source transformation. The task required substantial analysis of Java syntax and semantics, integration with the Eclipse Java front-end, and the necessary source code transformation and generation.

In other projects Rascal has been or is being applied to software metrics, static analysis (Java, PHP, Rascal), transformation (refactoring), code generation (Java, HTML), and domain specific language design (forensics, auditing, courseware). The design and implementation of Rascal are in an advanced stage but are not yet frozen. Based on these examples, we believe its integrated design is on the right track. The pieces of the puzzle are coming together.

Photo: Liza Daly, http://www.flickr.com/photos/lizadaly/2944376209/

The pieces of the puzzle are coming together.

Links:
http://www.cwi.nl/research-groups/Software-Analysis-and-Transformation
http://www.rascal-mpl.org/
http://www.springerlink.com/content/w60444612306qp54/fulltext.pdf
http://dx.doi.org/10.1145/1868281.1868291

Please contact:
Jurgen Vinju, CWI , The Netherlands
Tel: +31 20 592 4102
E-mail: Jurgen.Vinju@cwi.nl

An Environment for Dedicated Software Analysis Tools

by Muhammad Usman Bhatti, Nicolas Anquetil and Stéphane Ducasse

Moose is an open-source platform for the assessment of software and data analysis. Moose provides several engines to build tools, analyses and visualizations. One of Moose's strengths is the possibility to rapidly build domain-specific analysis tools.

To cope with a world in constant evolution, software systems must also evolve, otherwise they become progressively less relevant. Software evolution is a major contributor to the cost of software. Recent studies show that software evolution effort can represent 90% of the total software development effort. Of this 90%, between 40 and 60% is spent on code reading and program understanding. This is not surprising given that software systems are increasing in complexity every day.

To control these costs we need tools for code understanding and program assessment such as software metrics that provide an insight into software systems. But available tools are often too general, and inappropriate for specific domains or problems. We need intelligent tools to analyze systems because organizations have specific problems that can only be solved with dedicated solutions. To provide dedicated tools in a cost effective manner, we need environments for creating such tools. Moose is an extensive platform for software and data analysis rooted in the core idea of building dedicated tools.

Software and data analysis with Moose

Moose is a free and open-source project started in 1996 in the context of the FAMOOS European project to study evolving object-oriented systems. Since its conception it has spread to several research groups and it is increasingly being applied in industrial contexts. In total, the effort spent on Moose equates

to more than 150 person-years of research and development.

Moose offers standard solutions (quality metrics) but more importantly it allows rapid assessment of software quality and system structure through rapid building of small context-specific tools for very specific situations. Moose offers solutions that are specifically addressing the questions and concerns of daily software and its evolution.

Components in Moose

Data from various sources (source code, version control systems, bug tracker systems) and in various formats is imported in Moose. The imported data is stored in a model independent of the programming language. Supporting new data merely requires adapting the generic model and importing the data into the environment. Once information is imported, analysts can take advantage of the different tools for crafting software analysis tailored to their needs.

FAMIX is a language independent meta-model that can represent in a uniform way multiple object-oriented and procedural language entities or other software entities such as bugs or versions. All Moose components work on FAMIX models and as such are completely independent of the actual source from which the data were extracted.

Metrics: Moose provides software quality metrics for program assessment. Existing metrics include standard metrics such as cyclomatic complexity, depth of inheritance tree and so forth. Custom metrics suites, on all represented entities, can also be built on top of the information available.

Mondrian: A scripting engine that enables interactive crafting of custom visualizations. Typical visualizations include representing software components such as packages, classes, or methods and their dependencies. Visualization can be enriched by sizing and/or colouring nodes according to specific properties (see Figure 1). For example, the colour shade may depend on the value of a metric: the longer a method in the source code, the darker it will be in the visualization; similarly, the height and the width of nodes may depend on different metrics. Colour may also be used to highlight specific entities fulfilling a certain criterion, eg all entities from a specific author can be coloured in blue.

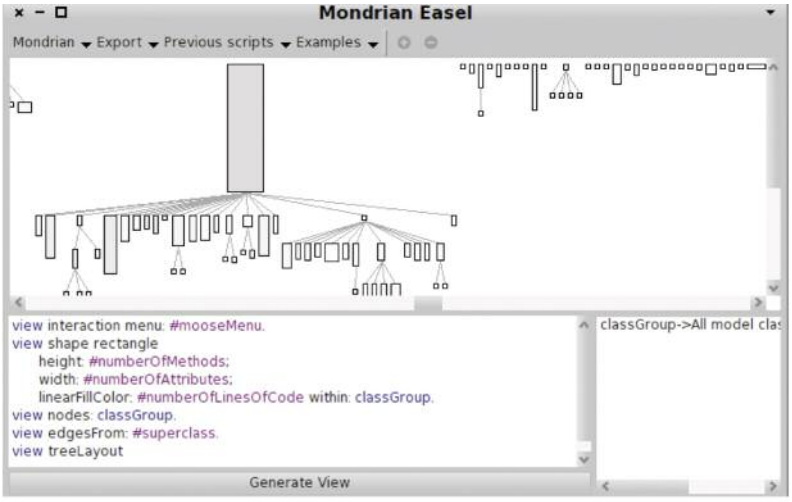


Figure 1: A visualization in Mondrian showing classes

Glamour: An engine for building dedicated browsers. It can be used to present software analysis in the form of custom reports or to create dedicated browsers. Through a simple model of information flow, different panes can present various points of view at various levels of detail on the data. For example, selecting a class (small square) in the top left pane of the visualization in Figure 2 will show metrics about it on the top right pane, its source code in the bottom left pane, and the classes of its package that have bugs in the bottom right pane.

Arki: An engine to describe and automatically verify rules on the entities of the model. For example one could specify that the database should not be called from the user interface layer, ensure architecture compliance, check specific code conventions, verify the quantity of comments, etc.

Conclusion

Software evolution is an activity that requires adapted solutions. Software analysis tools cannot act as universal oracles independent of the specific situation of the organizations using them. Hence, modern software analysis environments should provide services that allow users to express their requirements without taking too much effort. Moose allows for rapid assessment of software quality by rapid building of small context-specific tools for very specific situations.

Link:
The Moose Book:
http://www.themoosebook.org/

Please contact:
Usman Bhatti
Inria, France
Rmod project-team
E-mail: muhammad.bhatti@inria.fr

Figure 2: A visualization on bug incidence in classes

Recovering Software Architecture with Softwarenaut

by Mircea Lungu and Oscar Nierstrasz

As a complex software system evolves, its architecture typically degrades, and thus it becomes progressively harder to understand and maintain. Softwarenaut is an interactive and collaborative tool that helps developers recover and consequently improve the architecture of such systems by offering mechanisms to represent, filter, query and manipulate architectural views of the system under analysis. Softwarenaut is an open and extensible tool that is part of the Moose software analysis platform, which is used for numerous academic and industrial projects related to software evolution.

Architecture recovery is the process of recovering high-level views of a software system by analyzing the source code, the running system, and other available artifacts, such as the version history, bug reports and documentation. Architecture recovery faces two critical challenges: first, large amounts of static and dynamic information must be analyzed and processed, and second, the developers attempting to recover architecture typically have incomplete knowledge of the system they are analyzing. It is thus critical to have good tool support.

Softwarenaut tackles these challenges by offering high-level, graphical means to view software artifacts and their relationships, and mechanisms to navigate, filter, manipulate and share these views.

Figure 1 shows a screenshot of Softwarenaut being used to analyze ArgoUML, an open source UML editor. The Architectural View offers a high-level view of the modules in the system and the key relationships between them. The modules are represented as treemaps, and the size of each module is proportional to its actual size in lines of code; the width of each relationship is proportional to the number of invocations between the corresponding modules. A developer analyzing a system produces such an architectural view by iteratively applying filters to either remove certain types of modules and dependencies or to focus on those of interest. The views produced are interactive, allowing one to inspect details, or expand, collapse and zoom into individual modules and relationships. The views can also be saved and shared through a global architectural view repository such that different users analyzing the same version of a system benefit from previous work.

Evolutionary analysis is critical in implementing two core features of Softwarenaut: Filters and Inspectors:

Filters for relationships and modules are a mechanism for coping with the complex graphs that large systems entail. They display only those elements in an architectural view that are important for a given task and thus focus the analysis. There are different types of evolutionary filters that can be applied to both modules and relationships. Figure 1 shows only those relationships that were present in the six major releases of the system that we analyzed. This considerably reduced the number of displayed relationships.

The inspectors provide detailed information about a module or relationship.

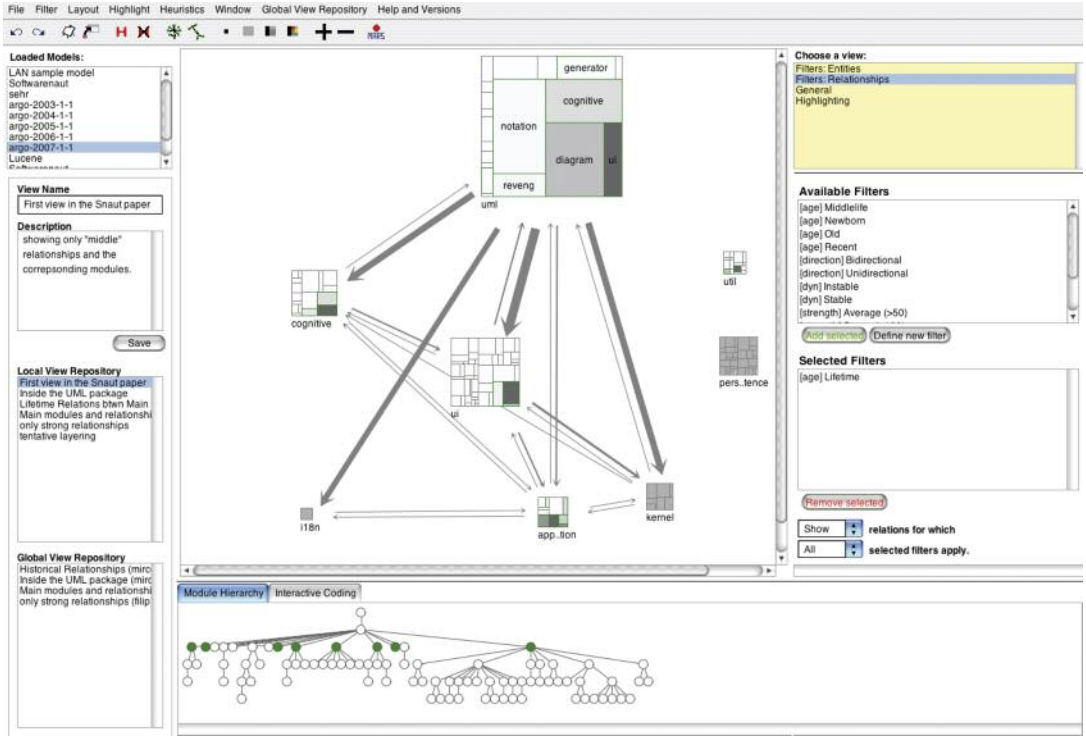


Figure 1: Using Softwarenaut to explore ArgoUML

One such example is the Module Evolution Filmstrip, which presents the evolution of a module in time. It is a visualization in which the versions of the module are presented chronologically from top to bottom. Each version is represented as a treemap inside which the contained classes are visible. The classes that are new in that version are highlighted with yellow and the classes that are modified are highlighted with a shade of blue proportional to the amount of changes.

Figure 2 presents the Evolution Filmstrip for the org.argouml.persistence module. In the first analyzed version it only contains two classes and the inspector shows that several classes replace them in the next version. By inspecting the names of the involved classes we discover that the two classes in the initial version that were responsible with interfacing with the database (DBReader and DBWriter) were replaced with classes that are responsible with file serialization (eg XMLInputStream).

One of our future research directions is to explore ways in which the recovered views can be integrated in the forward engineering process, support collaboration, and function as a live documentation of an evolving system. A view recovered for a given version of the system can function as a reference point for presenting the future evolution of the system. We also plan to run controlled experiments and user studies to evaluate efficiency and usability.

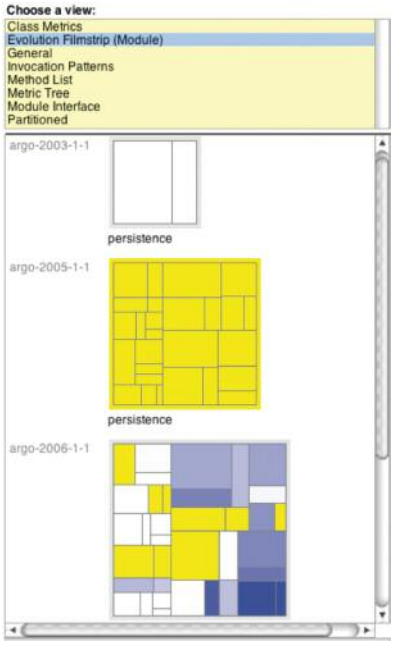


Figure 2: The Module Evolution Filmstrip of the org.argouml.persistence module

Softwarenaut was initially developed by Mircea Lungu at the REVEAL group at the University of Lugano, and is now being extended within the Software Composition Group at the University of Bern. The tool is freely available under the MIT license and is integrated in the Moose analysis platform. It has been used extensively for industrial consulting and teaching at both the universities of Bern and Lugano. We are interested in both academic and industry collaborations to further develop and assess Softwarenaut.

Links:
http://scg.unibe.ch/softwarenaut
http://www.moosetechnology.org
http://www.inf.usi.ch/faculty/lanza/reveal.html

Please contact:
Mircea Lungu
University of Bern, Switzerland
E-mail: lungu@iam.unibe.ch

Managing the Evolution of FOSS Systems

by Davide Di Ruscio, Patrizio Pelliccione and Alfonso Pierantonio

EVOSS, a model-driven approach that manages the upgrade of Free and Open Source Software (FOSS) systems, is presented. The approach simulates upgrades so that failures can be predicted before they affect the operating system. Both fine-grained static aspects (eg, configuration incoherencies) and dynamic aspects (eg, the execution of configuration scripts) are taken into account, improving over the state of the art of upgrade planners. Effectiveness is validated by instantiations with widely-used FOSS distributions.

FOSS distributions are composed of thousands of components (software packages) evolving rapidly and independently. A FOSS distribution is a consistent and functional collection of software packages comprising a complete operating system. The management of the evolution of FOSS distributions is very challenging due to the community-centric nature and the frequent release of components.

Distributions typically have automated mechanisms, called meta-installers, to manage their components and system evolution. Current meta-installers can successfully manage only a very limited set of upgrades. More precisely, current upgrade management tools (eg, the package managers in Linux distributions) are only aware of certain static dependencies that can influence upgrades. The most important information concerns the specification of inter-package relationships such as dependencies (ie, what a package needs in order to be correctly installed and function correctly), and conflicts (ie, what should not be present on the system in order to avoid malfunctioning). These tools completely ignore relevant dynamic aspects, such as potential faults of configuration scripts executed during upgrade deployment. It is not surprising that an apparently innocuous package upgrade can lead to a broken system state.

EVOSS (EVolution of free and Open Source Software), proposed within the context of the EC 7th framework project Mancoosi, is a model-based approach to support the upgrade of FOSS systems (see Figure 1).

In order to make upgrade prediction more accurate, EVOSS considers both static and dynamic aspects of upgrades. The main dynamic aspects considered are those related to the behaviour of package configuration scripts (main-

tainer scripts) which are executed during upgrade deployment.

Maintainer scripts are executed during upgrades and they are fully-fledged programs usually written in POSIX shell language. Moreover, they are run with system administrator rights and may perform arbitrary changes to the whole system. EVOSS defines a domain-specific language (DSL) to specify script behaviour: this is a way to limit the expressive power of the language used in maintainer scripts, without reducing the functionality of the scripts themselves. The DSL includes a set of high level clauses with a well-defined transformational semantics expressed in terms of system state modifications: each system state is given as a model and script behaviour is represented by corresponding model transformations.

The idea of EVOSS is to simulate system upgrades through its upgrade simulator component (see Figure 1), which allows system users to identify upgrade failures before performing the upgrade on the real system. The simulator takes into account both fine-grained static aspects (eg configuration incoherencies) and dynamic aspects (eg the execution of maintainer scripts). The simulator is based on model-driven techniques and makes use of a model-based description of the system to be upgraded. In order to build the system's configuration and package models, EVOSS makes use of model injectors that extract models from existing artifacts. The outcome of system injection is a model that represents, in a homogeneous form, different aspects of a running system, such as installed packages, users and groups, mime type handlers, alternatives, implicit dependencies, etc. The outcome of package injection contains modelling elements encoding both the package and its scripts (as DSL statements).

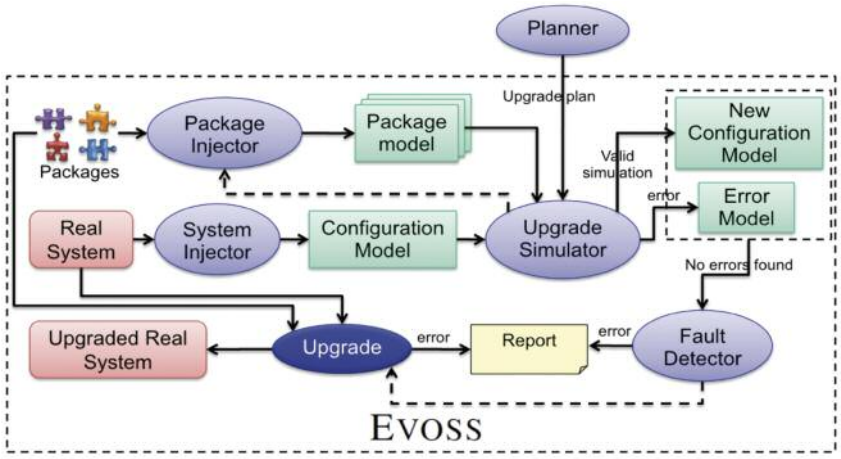


Figure 1: Overview of the EVOSS approach working in a real system

The fault detector is then used to check system configurations for incoherencies. The coherence of a configuration model is evaluated by means of queries which are embodied in the fault detector. Obviously it is not possible to define once-and-for-all a complete catalogue of faults because they are based on experience and acquired know-how. The fault detector has thus been designed to be open and extensible so that new queries can be added whenever new classes of faults are identified.

In summary, EVOSS represents an advancement, with respect to the state of the art of package managers, in the following aspects: (i) it provides a homogeneous representation of the whole system's configuration in terms of models, including relevant system elements that are currently not explicitly represented, (ii) it supports the upgrade simulation with the aim of discovering failures before they can affect the real system, (iii) it proposes a fault detector module able to discover problems on the configuration reached by the simulation.

Links:
http://www.mancoosi.org
http://evoss.di.univaq.it/

Please contact:
Davide Di Ruscio, Patrizio Pelliccione or Alfonso Pierantonio
University of Aquila, Italy
E-mail: davide.diruscio@univaq.it, patrizio.pelliccione@univaq.it, alfonso.pierantonio@univaq.it

Process Mining Software Repositories: Do Developers Work as Expected?

by Alexander Serebrenik, Wouter Poncin and Mark van den Brand

Modern software development commonly makes use of a multitude of software repositories. How can these help us to understand the on-going development process? Researchers of Eindhoven University of Technology design new methods revealing how software has been developed.

Modern software development commonly makes use of various configuration management systems, issue tracking systems, automated quality control and testing systems. Analysis of data in the repositories can give valuable insights into the development process of legacy systems. For example, it can answer the following software engineering questions: is the project development documentation kept up-to-date with the implementation? How fast are bugs resolved and feature requests implemented? Which (groups of) developers are responsible for the introduction of bugs or overtly complex code and code that does not meet company-specific and/or industry guidelines and standards such as MISRA? When and where have particular standard violations been introduced and (how) are

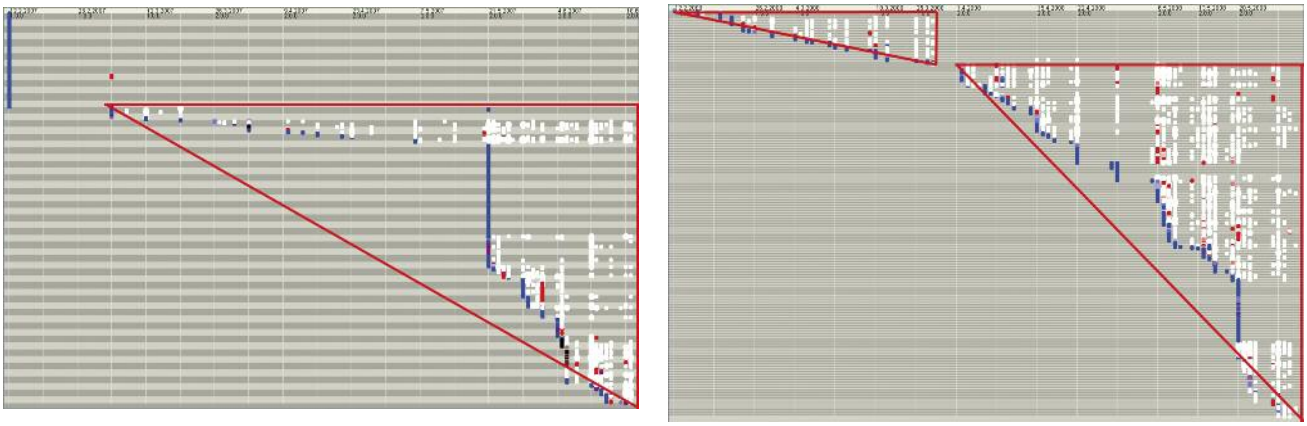


Figure 1: Visualizations of FRASR+ProM: the horizontal axes represents time, lines correspond to files, colored spots indicate events pertaining to the files such as creation (blue), modification (white) and deletion (red). Big red triangles have been added for readability purpose. The figure on the left shows only one triangle starting at the beginning of the project, ie, despite the prescribed development guideline, the prototype has been reused in the final implementation. The figure on the right shows two triangles: the triangle on the top corresponds to the prototype, the triangle on the bottom corresponds to the final implementation, ie, software development adhered to the prescribed guideline.

they related to later discovered bugs? If there is no relationship, is it worth narrowing the scope of routine quality checks to the actually important ones? What is the share of software artifacts covered by tests and how does this share change in time (eg from version to version) and project space (from subsystem to subsystem or one developer group to another)?

In the on-going project at Eindhoven University of Technology, we design a generic framework that allows the user to choose repositories, as well as analysis techniques depending on the software engineering question at hand. This is in sharp contrast with many existing repository mining approaches that consider a single specific question (eg which developers are responsible for introduction of bugs) or one specific kind of repository or group of repositories (usually a version control system or a version control system together with the bug tracker).

To create the generic framework we need to separate the preprocessing step, consisting of choosing and combining different repositories, from the analysis and visualization step. The first step takes care of specific challenges pertaining to the presence of multiple repositories, such as matching identities of developers and artifacts mentioned across different repositories, as well as synchronization of the time stamps. This step is carried out by a tool called FRASR. FRASR extracts from various data sources such as version control systems, bug trackers, mail archives, news groups, Twitter messages or issue reports, and combines this information in one log file. Given the log file, the second step implements a broad spectrum of analysis and visualization techniques supporting the user in answering the software engineering question. To implement the second step we make use of the existing work on process mining and ProM, a generic open-source framework for implementing process mining tools. Process mining aims at discovering, analyzing and visualizing business process models given a log of an information system. We stress that decoupling log preprocessing (FRASR) from the actual mining (ProM) makes a broad spectrum of analysis and visualization techniques readily available, and creates a highly flexible platform for application of repository mining.

We have successfully applied process mining software repositories to study a number of open-source software systems and student projects. We have considered such aspects of the development process as roles of different developers and the way bug reports are handled. In student projects we have focused on investigating whether the students adhered to the prescribed software development process. Using FRASR+ProM combination we have produced the following visualizations. For one of the projects the visualization clearly shows two triangles corresponding to the prototype implementation and the final implementation. For another project the visualization shows one triangle starting from the beginning of the project and continuing throughout, ie the prototype was reused as part of the final implementation delivered to the customer.

Process mining of software repositories provides a solid basis for prediction-based analysis. While the preceding questions aimed at providing insights into the way software and related documents have evolved so far, prediction-based analysis focuses on providing insights into the way software and related documents can or will evolve in the future. We consider developing appropriate prediction techniques as an important direction of the future work.

Process mining of software repositories is a novel promising approach that allows developers, designers and managers to rapidly gain insights in the way the development process is progressing, obstacles it is facing and challenges it has to address.

Link:
http://www.frasr.org/

Please contact:
Alexander Serebrenik, Eindhoven University of Technology,
The Netherlands
Tel: +31 402473595
E-mail: a.serebrenik@tue.nl

Internet Software Evolution with VPraxis

by Xavier Blanc and Jean-Rémy Falleri

VPraxis is a tool to support the evolution of Internet applications. It automates the construction of a software application's history together with the histories of used resources. Comparing these histories enables change impact analysis and eases evolution management.

Nowadays, almost all software applications, regardless of their purpose or domain, run on the Internet. Applications such as eCommerce, social networks, telecommuting platforms and business to business market places run on the Internet. Some are so tightly coupled with the internet that without an internet connection they would not survive.

An application that runs on the internet belongs to a complex ecosystem. It uses and provides resources that compose the ecosystem. Those resources, which were as simple as pictures or HTML documents a few years ago, are now highly complex such as web services or web semantics data. Moreover, the ecosystem is becoming increasingly dynamic as resources change every day. This makes classical software development inefficient, where well-structured teams apply traditional development cycles to build all the software artifacts that compose an application.

As shown by Baskerville, classical software development is inadequate to Internet-speed software development. Traditional principles such as "reuse" or "quality" have no place in such a context. Quality, which is expressed as a requirement and validated throughout the development cycle in a traditional development process, is on the contrary, continuously negotiated with the client and subjectively validated by the users in case of Internet applications. Reuse is not even an option as software components are deprecated after just a few months of life on the Internet.

New development approaches have been proposed to face the challenges of Internet application development. For instance, agile methods propose mechanisms to deal with continuous requirements. Web frameworks such as Ruby On Rails or Zope integrate new Internet technologies. However, even if these approaches do help development, there are still few solutions to tackle evolution. This is one factor contributing to rising maintenance costs.

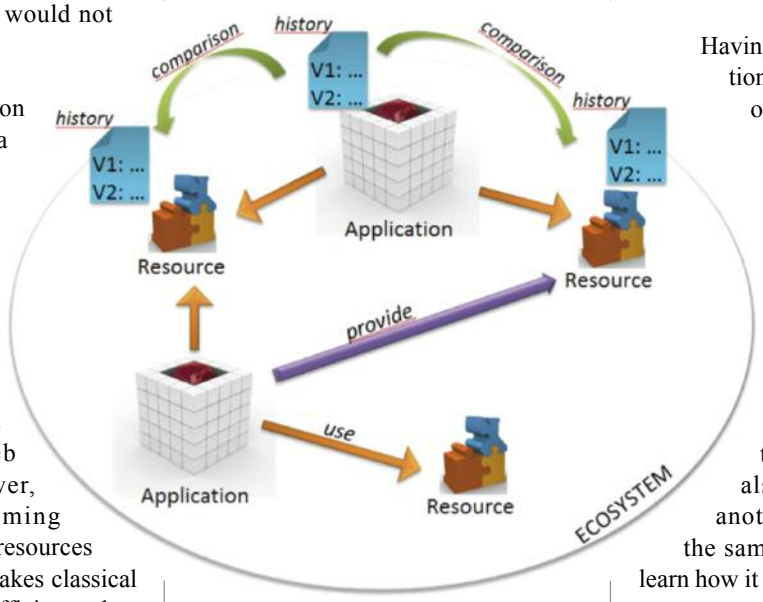


Figure 1: An evolving software ecosystem monitored by Praxis histories.

Mastering the evolution of an Internet application requires being aware of the evolution of the ecosystem, or at least of the evolution of the resources used by the application. This requires answers to the following questions: "How to monitor the evolution of all used resources?", "How to measure the impact of their evolutions?", "How to know when to evolve?" and "How to identify which parts of an application have to evolve?". The purpose of our research is to answer these questions in order to automate operations performed during evolution.

Our approach consists of uniformly capturing the evolution of any application and of any used resources. We base our approach at a structural level since most of the broken evolutions occur at a structural level. We propose VPraxis, an operation-based model that unifies both source code and evolution information. Within this model, the various software artifacts that compose an application or that describe a resource are described by a history. A history contains all the actions performed to create and modify software artifacts.

Having the history of an application together with the histories of the resources it uses allows for comparison of histories. For example, with the history of an Internet application and the histories of the web services it uses, we can detect that one of the proposed services has changed and an evolution of the application is therefore required. We can also analyze the history of another application that uses the same web service in order to learn how it has evolved.

We are already able to automatically compute histories. When a versioning system is available, a history can be automatically constructed by analyzing the source code it manages. We have developed a tool that automatically constructs the history of projects managed by SVN versioning systems. In our current implementation, we support Java, CSS, HTML and XML artifacts. A history contains all operations performed to edit these artifacts since the beginning of the project. We can also build a history of a used resource by browsing its description daily, as a web crawler would do. For instance, we can build the history of web services by browsing their WSDL descriptions.

We have already performed some comparisons of histories. We have based our analysis on logical rules. For instance, we have defined a logical rule specifying that any called service has to be described by the WSDL. We then provide incremental analysis that checks

the rule each time the history of the application or the history of the web service is updated. If the service is no longer supported, the check of the rule reports that an evolution is needed.

As a synthesis, the goal of our approach is to support the evolution of Internet applications by monitoring changes performed to the ecosystem. We propose mechanisms to capture the evolution of software artifacts scoped to a given application. Thanks to our concept of history, we provide reactive local views of evolution. We use those local views to compute analysis, such as comparison of histories, in order to automate maintenance operation.

Links:
http://sphere.labri.fr
http://code.google.com/p/harmony/

Please contact:
Xavier Blanc and Jean-Rémy Falleri
University Bordeaux, LaBRI, France
E-mail: xavier.blanc@labri.fr,
jean-remy.falleri@labri.fr

Open-Source Formative Evaluation Process in Remote Software Maintenance

by Javier Cano, Christophe Joubert, Miguel Llácer and Miguel Montesinos

One of the main challenges of software evolution is to provide software applications with a maintenance environment with maximum time efficiency at minimum cost and the strongest accuracy. This includes managerial aspects of software evolution, like effort estimation, prediction models, and software processes. In this work, we focused on formative evaluation processes to give early feedback on the evolution of development metrics, global quantitative goals, non-functional evaluation, compliance with user requirements, and pilot applications results. Our results were applied to a monitoring control platform for remote software maintenance.

In the European project FASTFIX (ICT-258109, June 2010-November 2012), a monitoring control platform for remote software maintenance is being developed. Within this project, we have been working on a formative evaluation process that evaluates methods, concepts and prototypes as the project is being developed. This process allows for a continuous integration and enhancement of the platform. As a result, the development process is steadily monitored and evaluated, and early feedback is provided to the designers, developers and architects. The metrics on which the formative evaluation process is based are described in Table 1. Targets are monitored by several open source tools on top of the developed codebase and industrial trials.

The formative evaluation process that we have defined can be split in three stages as described in Figure 1. Each stage is composed of several open-source tools for the incremental and continuous verification and validation of non-functional aspects of evolving software. The overall benefit of the process is to correct problems and to include new requirements before the production phase of the software project and during the whole life of the project.

Following an agile development process (Scrum), the process starts with incremental internal releases that are produced based on the requirements specified for the platform. Tools and methods to monitor the development process have been chosen and integrated. Continuous construction and testing from subversion commits are supported through the Hudson platform, while systematic analysis of code metrics is realized through the quality management platform Sonar that reports on release quality. Only working qualified versions are issued and sent to the trial integration stage. The project roadmap aligns internal releases with integration tests to be executed after each milestone.

Following each internal release, there is an integration trial with an industrial application in order to provide early feedback. A complete plan of trials has been set up, including two industrial trials on top of the final platform to provide a measurement of the introduced global improvements. Tools and methods to obtain metrics have been chosen and deployed. The JMeter platform collects information about program execution and user interaction. As such, it allows identification of symptoms of safety errors during execution, critical errors like performance degradation or changes in user behaviour.

Finally, the data gathered during the trial after each internal release provides feedback for the development process on issues and new requirements. Feedback strategies are defined and management tools for them are deployed, in particular the Trac issue management system. This allows tight control of product families and software versioning.

Trials for the final evaluation of the platform are defined and they are used as an input for the summative evaluation

Monitorization target	Monitoring tool
Source code quality.	Static code analysis.
Components integration to form the FastFix platform.	Continuous integration.
Tests execution.	Automated builds and tests execution.
Platform integration in target applications.	Integration trials after internal releases.
Open and resolved issues.	Issues statistics reports.

Table 1: Formative Evaluation Metrics

process. They cover both quantitative and qualitative metrics.

The results to date are very encouraging: the formative evaluation process can be integrated within any software product lifecycle from early analysis stages to final maintenance stages; it gives feedback on the evolution of development metrics, global quantitative goals, non-functional evaluation, compliance with user requirements, and pilot application results.

In our research, we collaborated with the Irish Software Engineering Research Center (LERO, Ireland), the Technische Universitaet Muenchen (TUM, Germany), the Instituto de Engenharia de Sistemas e Computadores, Investigaçao e desenvolvimento in Lisbon (INESC-ID, Portugal), and the companies TXT E-Solutions SPA (Milan, Italy) and S2 Grupo (Valencia, Spain).

This research is also part of a horizontal task with another ICT Software and Service Architectures, Infrastructure and Engineering project, called FITTEST (Future Internet Testing), that deals with performance and efficiency issues related to the monitoring activity, and with fault correlation and replication techniques. Our work is partially supported by the Spanish MEC INNCORPORA-PTQ 2011 program.

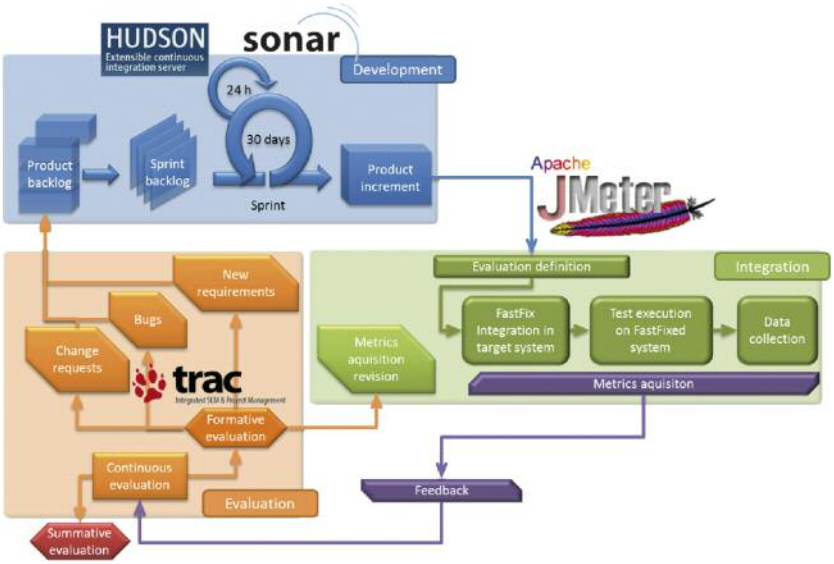


Figure 1: Formative Evaluation Process

Link:
http://www.fastfixproject.eu

Please contact:
Javier Cano
Prodevelop, Spain
Tel: +34 963 510 612
E-mail: fjcano@prodevelop.es

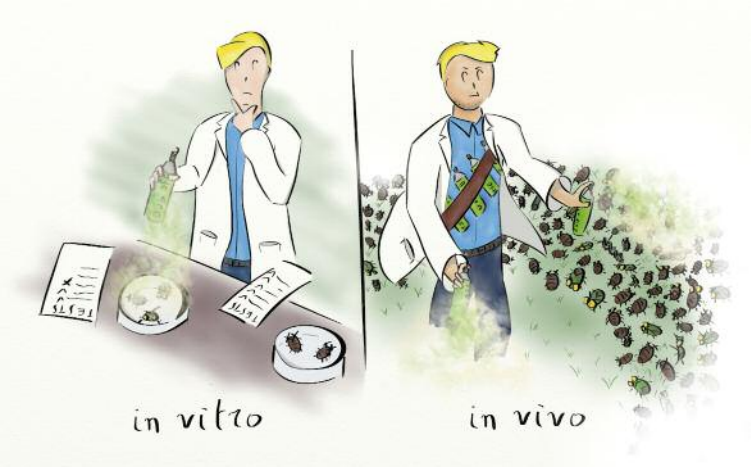
"in vivo" Research in Software Evolution

by Serge Demeyer, Ahmed Lamkanfi and Quinten Soetens

The production of reliable software is an important area of investigation within the software engineering community. For many years, reliable software was seen as software "without bugs". Today however, reliable software has come to mean "easy to adapt" because of the constant pressure to change.

As a consequence of the need to be adaptable, organizations producing software intensive systems must strive for a delicate balance between two opposing forces: the need for reliability and the need for agility. Regarding the former, organizations optimize for perfection; regarding the latter, they optimize for development speed. The ANSYMO research group at the University of Antwerp (Belgium) is investigating ways to reduce the tension between reliability and agility. We seek to bring the changes made to a software system into concrete entities that can be manipulated and analysed. With this method we expect to speed up the release process without sacrificing the safety net of quality control. A few examples follow.

- *Misclassified bug reports.* Text-mining techniques (such as those employed in search engines on the web) classify reported bugs into two categories: severe and non-severe. Severe bugs cause system crashes, hence software is seldom released when it is known that such bugs are lurking inside. On the other hand, non-severe bugs cause less harm; consequently it is common practice to release systems with a list of known (non-critical) bugs. To avoid bug-reports being assigned to the wrong category, we have identified text-mining techniques that can verify whether a reported bug is indeed severe or not.



- *What to (re)test?* To minimize bug resolution time, it is good practice to have a fine-grained suite of unit tests that is run frequently. If a small change to a program causes a test to fail, then the location of the defect is immediately clear and it can be resolved right away. Unfortunately, for large systems the complete suite of unit tests runs for several hours, hence software engineers tend to delay the testing to a batch mode. In that case if one unit test fails the cause of the defect is unclear and valuable time is lost trying to identify the precise location of a defect. With our research, we are able to identify which unit tests are affected by a given change, which enables us to run the respective unit tests in the background while the software engineer is programming.

"In Vitro" Vs. "In Vivo" Research

In this kind of research, industrial collaboration is a great asset. In the same way that a biologist must observe species under realistic circumstances, so must computer scientists observe software engineers in the process of building software systems. Industrial collaboration therefore shouldn't be seen as a symptom of "applied" research but rather be viewed as fundamental research in software engineering practices.

To stress the fundamental research angle, we adopt the terms "in vitro" and "in vivo". Indeed, just like in life-sciences, we take methods and tools that have proven their virtue in artificial lab conditions ("in-vitro research") and apply them in uncontrolled, realistic circumstances ("in-vivo research"). This can sometimes make a big difference. Referring back to the misclassified bug reports mentioned earlier, we originally designed the experiment to verify whether text-mining algorithms could predict the severity of new bug reports. However, from discussions with software engineers in bug triage teams, we learnt that this wouldn't be the best possible use case for such an algorithm because they rarely face problems with incoming bug reports. Also, close to the release date, pressure builds up and it is then they want to verify whether the remaining bug reports are classified correctly. Such observations can only be made by talking to real software development teams; mailing lists and software repositories are a poor substitute.

Luckily our research group has a strong reputation in this regard. Indeed, our PhD students actively collaborate with software engineers in both large and small organisations producing software intensive systems. As such we remain in close contact with the current state of the practice and match it against the upcoming state of the art.

Link:
http://ansymo.ua.ac.be/

Please contact:
Serge Demeyer, Ahmed Lamkanfi and Quinten Soetens
ANSYMO research group
University of Antwerp, Belgium
E-mail: serge.demeyer@ua.ac.be

Seeing the Forest for the Trees with New Econometric Aggregation Techniques

Alexander Serebrenik, Mark van den Brand and Bogdan Vasilescu

Understanding software maintainability often involves calculation of metric values at a micro-level of methods and classes. Building on these micro-level measurements and complementary to them, econometric techniques help to provide a more precise understanding of the maintainability of software systems in general.

Maintaining a software system is like renovating a house: it usually takes longer and costs more than planned. Like a house owner preparing a condition report identifying potential problems before renovation, a software owner should assess maintainability of software before renovating or extending it. To measure software maintainability one often applies software metrics, associating software artifacts with numerical values. Unfortunately, advanced software metrics are commonly measured at a level of small artifacts, eg methods and classes, and fail to provide an adequate picture of the system maintainability. Continuing the analogy, the state-of-the-art in software metrics corresponds to a condition report detailing the state of every brick and obscuring the general assessment in the multitude of details. Metrics visualization techniques provide a general overview of the measurements but have difficulties with presenting evolution of these measurements in time.

To see the forest of a software system for the trees of individual measurements, aggregation techniques are used. Current aggregation techniques are, however, usually unreliable or involve human judgment. For instance, the mean is known to become unreliable in the presence of highly skewed distributions, which are typical for software metrics. Another approach, distribution fitting, consists of manually selecting a known family of distributions and fitting its parameters to approximate the observed metric values. The fitted parameters can then be seen as the aggregation results. However, the fitting process should be repeated with each new metric considered. Moreover, it is still a matter of controversy whether, for instance, software size is distributed log-normally or double Pareto. Finally, threshold-based approaches assume the availability of a set of metric thresholds: depending on which thresholds the metric value of an individual element (eg method or class) exceeds, the value is classified as being poor, mediocre or good. If this is the case, the entire system is labelled as being poor, mediocre or good depending on the percentage of poor, mediocre and good values. By definition, threshold-based approaches require the presence of either commonly accepted thresholds

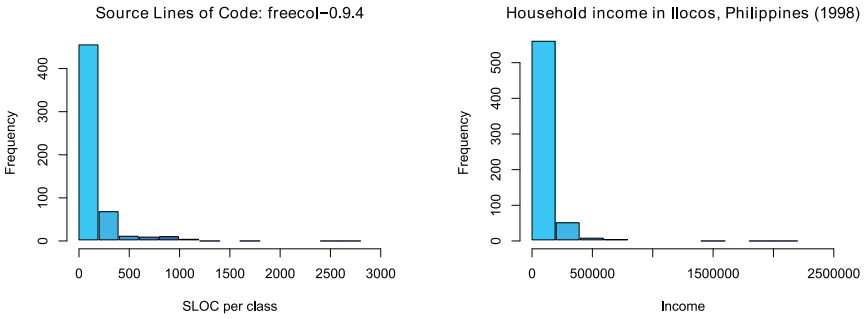


Figure 1: Software metrics (SLOC) and econometric variables (household income in the Ilocos region, the Philippines) have distribution with similar shapes.

or a large benchmark collection to derive such thresholds. Unfortunately, neither the public thresholds nor the composition of the benchmark collections are commonly accepted or can be scientifically validated. Furthermore, the threshold approaches assume that if all individual values are good, then so is the quality of the entire system. This, however, is not the case for such metrics as the depth of inheritance tree.

In an on-going project that commenced in 2010 a research team from Eindhoven University of Technology, The Netherlands (Alexander Serebrenik, Mark van den Brand and Bogdan Vasilescu) is investigating application of econometric inequality indices as aggregation techniques. Inequality indices are econometric techniques designed to measure and explain inequality of income or welfare distributions. Their application to aggregation of software metrics is based on the observation that numerous countries have few rich and many poor, and similarly, numerous software systems have few very big or complex components and many small or simple ones. Inequality indices combine the advantages of the preceding aggregation techniques and avoid their disadvantages. Similarly to distribution fitting, they provide reliable results for highly-skewed distributions. Similarly to the mean they do not require complex application procedures. Moreover, application of inequality indices does not require the availability of well-established thresholds or extensive benchmarking.

In 2010 we advocated the use of the Theil index, one of the econometric inequality indices, to measure and explain inequality among software metric values. We observed that inequality in file sizes of the Linux Debian distribution lenny can be better explained by the package these files belong to, rather than the implementation language. This suggests that if one would like to reduce this inequality, ie distribute functionality across the units in a more egalitarian way, one should focus on establishing cross-package size guidelines. In 2011 we conducted an extensive empirical comparison of different inequality indices and determined guidelines when different inequality indices should be used for software metrics aggregation. In another recent work we have applied inequality indices to studying the impact of different software project attributes on the development effort. We found that such project attributes as the primary programming language, the organization type, and the year of the project have a higher impact than the development platform or the intended market.

Our current research, carried out in cooperation with Inria, Université de Bordeaux and University of Paris 8 (France), compares inequality indices with a threshold-based technique called Squale. We have found a close theoretical relationship between the two, and we have further conducted an empirical evaluation of the approaches proposed. Even more recently, together with the colleagues from Software Engineering Lab of University of Mons (Belgium), we have been investigating the application of inequality indices to studying developer specialization in open-source projects.

As the most important direction of the future work we consider integration of the inequality indices proposed in predictive models. We believe that the ability of inequality indices to explain inequality of the metric values observed will provide for more precise prediction of the metric values.

Please contact:
Alexander Serebrenik
Eindhoven University of Technology,
Eindhoven, The Netherlands
Tel: +31402473595
E-mail: a.serebrenik@tue.nl

Continuous Architecture Evaluation

by Eric Bouwers and Arie van Deursen

Keeping the implementation of a software system in-line with the original design of the system is major challenge in developing and maintaining a system. One way of dealing with this issue is to use metrics to quantify important aspects of the architecture of a system and track the evolution of these metrics over time. We are working towards extending the set of available architecture metrics by developing and validating new metrics aimed at quantifying specific quality characteristics of implemented software architectures.

Most software systems start out with a designed architecture, which documents important aspects of the design such as the responsibilities of each of the main components and the way these components interact. Ideally, the implemented architecture of the system corresponds exactly with the design. Unfortunately, in practice we often see that the original design is not reflected in the implementation. Common deviations are more (or fewer) components, undefined dependencies between components and components implementing unexpected functionality.

Such discrepancies can be prevented by regularly evaluating both the designed and implemented architecture. By involving the developers as well as the architects in these evaluations, the implemented and the designed architecture can evolve together. Many evaluation methods are available, varying greatly in depth, scope and required resources. A valuable outcome of such an evaluation is an up-to-date overview of the implemented architecture and the corresponding design.

But when should such an evaluation take place? Depending on the resources required the evaluation can take place once or twice during a project, or periodically (for example every month). Unfortunately, in between the evaluations issues can still arise which may lead to deviations between the design and the implementation. The later these deviations are discovered the more costly it is to fix them.

One way to deal with these problems is to continuously monitor important aspects of the implemented architecture by the means of software metrics. Basic metrics, for example the number of components, are straight-forward to calculate after each change and can be used as a trigger for performing sanity checks as soon as the metric is changing outside its expected bounds. If the sanity check fails (ie when the change is incorrect) a more detailed evaluation should be performed, potentially leading to a full-scale architecture evaluation. See Figure 1 for an illustration of this process.

Basic metrics (number of modules, number of connections) are easy to measure and provide relevant information. However, an examination of these two metrics in isolation does not allow for detection of all types of change, for example when a single component is implementing too much of the overall functionality.

In our current research project we are extending the set of available architecture metrics with the addition of new metrics which are related to quality aspects as defined in ISO-9126. More specifically, we have designed and validated two new metrics which quantify the Analyzability and the Stability of an implemented software architecture.

The first is called "Component Balance". This metric takes into account the number of components as well as the relative sizes of the components. Owing to the combination of these two properties, both systems with a large number of components (or just a few components) as well as systems in which one or two components contain most of the functionality of the system receive a low score. We validated this metric against the assessment of experts in the field of software quality assessments by means of interviews and a case study. The overall result is a metric which is easy to explain, can be measured automatically and can therefore be used as a signaling mechanism for either light-weight or more involved architecture evaluations.

The other concept we introduced is the "Dependency Profile". For this profile each module (ie source-file or class) is assigned to one of four categories:

- modules hidden inside a component
- modules that are part of the required interface of a component
- modules that are part of the provided interface of a component
- modules that are part of both the required and the provided interface of a component.

The summation of all sizes of the modules within a category provides a system-level quantification of the encapsulation of a software system. This metric has been validated by an empirical experiment in which the changes which occurred within a system are correlated to the values of each of the four categories. The main conclusion of the experiment is that with more code encapsulated within the components of a system, more of the changes remain localized to a single component, thus providing a measure for the level of encapsulation of a system which can be calculated on a single snapshot of the system.

Both of the metrics have shown to be useful in isolation. We are taking the next step by determining how these metrics can best be combined in order to reach a well-balanced evaluation of an implemented architecture. In order to answer the question when a more elaborate evaluation should take place we are planning to determine appropriate thresholds for these two metrics. The combined results of these studies will ensure that these metrics can be embedded within new and existing processes for architecture evaluations.

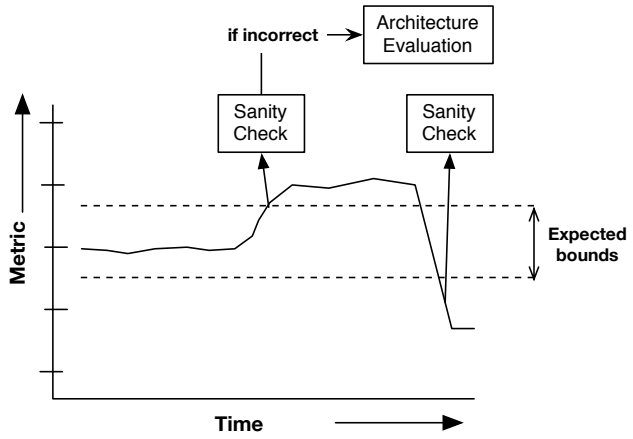


Figure 1: The process of continuous architecture evaluations by the means of metrics. Simple metrics characterizing the architecture are measured over time and as soon as a large deviation occurs a sanity check is performed, potentially leading to a full-scale architecture evaluation if the change in the metric is incorrect.

Links:

http://www.sig.eu/en/Research/Key%20publications/Archive/923.html

http://www.sig.eu/nl/Research/Wetenschappelijke%20publicaties/1023/__Dependency_Profiles_for_Software_Architecture_Evaluations__.html

Please contact:

Eric Bouwers,
Software Improvement Group,
The Netherlands
Tel:+31 20 314 09 50,
E-mail: e.bouwers@sig.eu

Arie van Deursen
Delft Technical University,
The Netherlands
E-mail: Arie.vanDeursen@tudelft.nl

Evolutionary Architecting of Software-Intensive Systems

by Jakob Axelsson

Most industrial software-intensive systems have a very long life span, and undergo many changes after their initial conception. To ensure that they maintain the desired system-level properties, it is essential that their architecture is also updated to reflect the added functionality. To this end, we have investigated how Evolutionary Architecting should be carried out in order to provide a strong basis for a system, not only at conception but throughout its life. The results are concrete and based on empirical findings, and include both a new state-of-the-art process description and a light-weight maturity evaluation method.

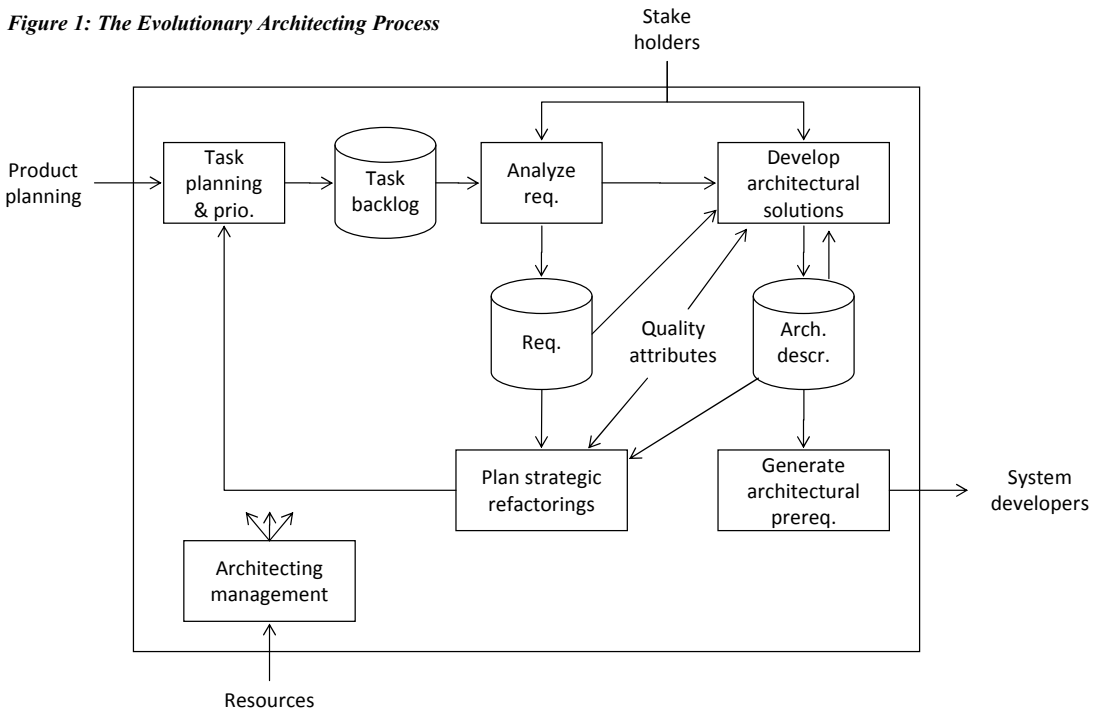
Embedded software is a critical component to be considered in the development of many new products such as cars, airplanes, industrial robots, and telecommunications systems. There is great scope for innovation within software, and innnovative software can keep a product competitive. Often, a product line approach is used, where many variants are spawned off the same platform over decades. However, with separate teams working on different products and modifying the platform to suit their needs, coordination between them becomes essential to ensure that the platform keeps its integrity and stays useful for future needs. It is a central role of the architects to maintain this integrity and provide technical prerequisites for the different design teams.

Plenty of research has been done on software and systems architecture, but much of it focuses on the development of new systems. However, our empirical research shows that evolution of existing products is much more common than new development. Therefore, through interviews and surveys with around 100 active architects at a dozen companies, we have tried to identify needs and challenges within architecting to support the evolutionary development of embedded software. Many of the organizations suffer from unclear processes and responsibility for architecture, and lack methods to evaluate the business value when making architectural decisions. In short, the companies rely on individuals instead of on processes and methods.

These issues are typical signs of immaturity in architecting organizations. To identify where improvements can be made, it is useful to have a way of assessing an organization's maturity. Several well-known maturity models exist, such as Capability Maturity Model Integration (CMMI), which captures the best practices for development in general. However, it has a few drawbacks when applied to architecting. Firstly, it is heavy to use for a small team, and secondly, it emphasizes formal appraisals by external assessors, whereas architects are more in need of self-assessment. As an alternative, we have developed an Evolutionary Architecting Maturity Model (EAMM) which is an instantiation and simplification of CMMI. The appraisals in EAMM are extremely simple, with just 53 questions taking about one hour to complete, but it still pinpoints where the organization should focus on making improvements.

We have applied EAMM at a handful of industrial companies, and a typical company ranks between level two and three on the five-level maturity scale used by EAMM and CMMI. Specific areas that are often weak include management of architectural requirements, quality assurance, verification, measurement and analysis, and risk management. It is

Figure 1: The Evolutionary Architecting Process



striking that many companies lack a written description of their architecting process, and this is a fundamental flaw if one wants to work systematically with process improvement. To address this, we have devised an Evolutionary Architecting Process (EAP) which can be used as a starting point for a company to improve its architecting practices.

The top level EAP process is shown in Figure 1. The process starts with a change request from the product planners. These requests are planned and prioritized and placed in a backlog. The architects then start a task by analyzing the requirements to find out which of these are significant for the architecture. This involves close interaction with various stakeholders, and the final requirements are stored in a database for use in future iterations. Based on the requirements and on quality attributes of the architecture, architectural solutions are developed and evaluated. From the selected option, prerequisites are generated that can be used by system developers for subsequent detailed design and implementation.

Quality attributes play an important role, and include factors such as flexibility, scalability, safety, and understandability. Maintaining these attributes, while dealing with the specific requirements of a particular function, is at the core of architecting. Sometimes it is also necessary to induce specific activities called refactorings that do not alter the functionality of the system but change its internal structure to improve the quality attributes.

With the development of the EAMM and EAP, we provide a new foundation for the kind of architecting that is most important for the software-intensive systems industry. In the future, we will further investigate and refine these methods in close co-operation with partner companies.

Link:
http://www.mrtc.mdh.se/index.php?choice=staff&id=0148

Please contact:
Jakob Axelsson
SICS and Mälardalen University, Sweden
Tel: +46 72 734 29 52
E-mail: jakob.axelsson@sics.se

Automated Synthesis of CONNECTors to support Software Evolution

by Amel Bennaceur, Paola Inverardi, Valérie Issarny and Romina Spalazzese

Today's software systems are increasingly networked and are further characterized by ever-changing functionalities provided to/required from the networked environment because of the wide variety of dynamically available heterogeneous applications. To manage evolution in this context, the CONNECT project pursues the automated synthesis of CONNECTors enabling continuous composition and interoperability of applications.

Software evolution is a hot topic in today's distributed software systems, which are characterized by an increasing level of heterogeneity and dynamism. For instance, updated versions of software are continually made available, new applications are created and dynamically join the networked environment, old or broken applications are unavailable or removed, and new needs emerge.

Evolution management in this environment raises the need for novel computing paradigms able to ensure continuous run-time composition and interoperability of heterogeneous and dynamically available applications in an automated manner.

Although in principle they could interact since they have compatible (ie complementary) functionalities, heterogeneous applications populating this landscape can be characterized by discrepancies that may undermine their ability to seamlessly interoperate (ie communicate and coordinate). Discrepancies include incompatible interaction protocols and/or different interfaces meaning different actions and/or data models.

Examples of heterogeneous applications are shown in Figure 1(a) where the light blue entities show independently developed clients and servers for the organization of trips. The server exhibits separate searches for flights and hotels but the client would expect to have a single search with all the information.

To enable perpetual and automated composition and interoperability, the ICT FET IP European project CONNECT pursues the automated synthesis of CONNECTors that overcome interoperability barriers. This responds to the evolution of functionalities provided to and required from the networked environment of today's software systems. Automatically synthesized CONNECTors are concrete emergent entities that mediate the application discrepancies, ie translate and coordinate mismatching interaction protocols, actions, and/or data models, letting applications interact effectively. Examples of CONNECTors are illustrated in Figure 1(b) -green entities.

Automated Synthesis of CONNECTors

We have developed a theory for the automated synthesis of application-layer CONNECTors in order to formally ground and automatically reason on the interoperability problem among heterogeneous applications. The theory has also served as a principled basis for the development of a prototype tool that automatically generates CONNECTors.

A Theory of CONNECTors

Our theory for the automated synthesis of CONNECTors assumes the descriptions of two Networked Systems (NSs) focusing on their application-layer pro-

tocol. It then defines a reasoning process on the protocols provided that produces as output the behaviour of a CONNECTor that lets heterogeneous applications interact by mediating their discrepancies.

Specifically, the description of an NS encompasses the following models:

- interface definition, in terms of actions required and provided by the NS, together with associated input/output data
- interaction protocol specification, ie the behavioural automaton describing the process according to which actions from the interfaces can be invoked
- ontological description of actions and data which refers to concepts of a widely shared application domain ontology. This offers a means to automatically and dynamically reason on the semantic interoperability of applications for which we do not have a priori knowledge.

The CONNECTor theory reasoning decomposes into three phases or steps: Abstraction, Matching and Mapping.

1. Abstraction makes the Networked System models comparable and, if possible, reduces their size thus simplifying and speeding up the reasoning on them.

2. Matching checks the Networked Systems compatibility identifying possible interaction protocols incompatibility and/or action differences and/or data model differences preventing seamless interoperation.

3. Mapping (or Synthesis). A successful matching check identifies possible discrepancies preventing proper coordination of networked applications. Mapping solves such mismatches, leading to the identification of the existence of a CONNECTor. Consequently, this step produces a proper CONNECTor that interposes between application protocols and, while managing heterogeneity, allows applications to effectively exchange compatible sequences of actions with data.

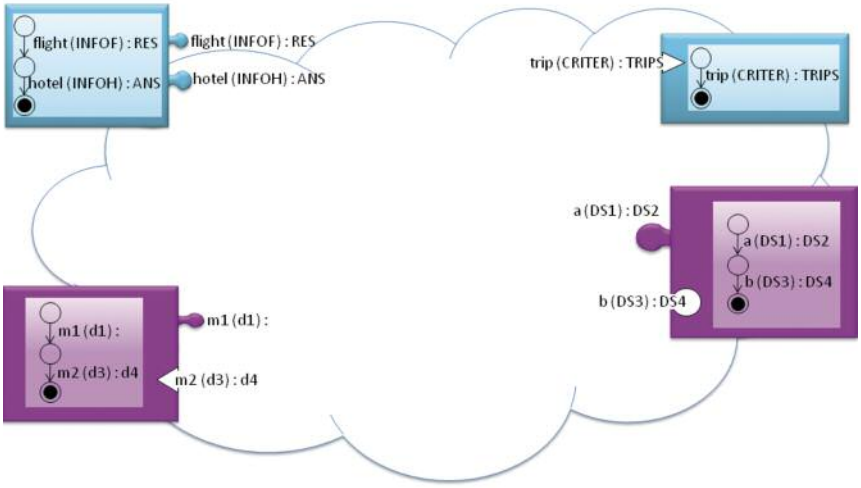


Figure 1(a): Examples of heterogeneous networked applications

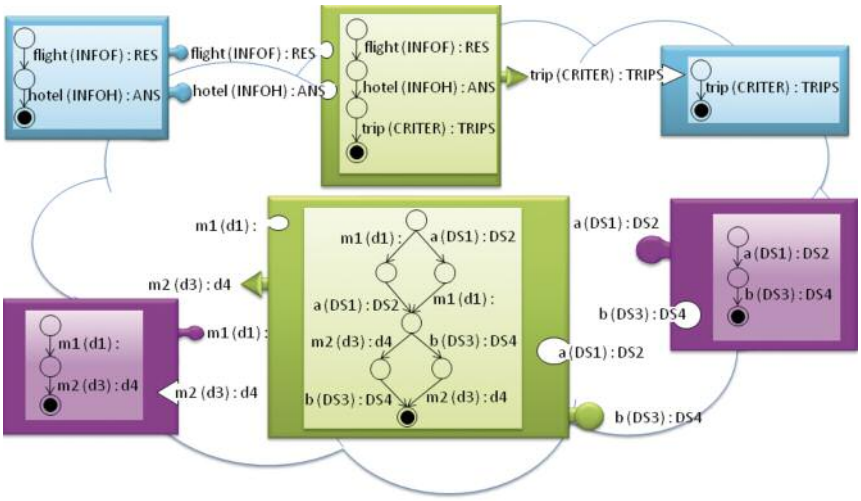


Figure 1(b): Example of CONNECTors enabling composition and interoperability in the heterogeneous networking environment

Future perspectives

This article has outlined our theory of CONNECTors that supports software evolution and more specifically eternal interoperability of networked systems through on-the-fly synthesis of CONNECTors.

This work is part of the larger CONNECT project effort, that revisits the middleware paradigm to face the challenges raised by increasingly dynamic and heterogeneous distributed systems. Current distributed systems exhibit a far greater complexity than that tackled by state of the art middleware; this will be even truer in the future. Hence, the project focuses on sustaining interoperability in an ever diverse and continuously changing networking environment. As a result, CONNECT introduces the paradigm of dynamically created "emergent middleware", based on supporting enablers for protocol synthesis and also learning. Emergent middleware is further grounded on the emerging models@runtime paradigm due to the extensive use of networked systems models at runtime to be able to learn, reason about, and also compose the protocols that are executed.

Links:

http://www.connect-forever.eu/
http://www.di.univaq.it/romina.spalazzese/publications.html

Please contact:
Romina Spalazzese
University of L'Aquila, Italy
E-mail: romina.spalazzese@univaq.it, romina.spalazzese@gmail.com

Valérie Issarny
Inria, France
Arles project-team
E-mail: Valerie.Issarny@inria.fr

Emergent Middleware

by Paul Grace, Gordon S. Blair and Valerie Issarny

We are moving towards a world in which everything is connected. Yet such a goal highlights the deficiencies of today's systems platforms in achieving a fundamental property of distributed systems, namely interoperability. Faced with extreme heterogeneity of computational devices and networks, how can we ensure that every system can talk to every other system?

Interoperability is the ability for two systems to exchange, understand and use each other's data, and is a long-standing problem in the field of distributed systems. However, the emergence of pervasive computing and the Internet of Things have brought about new challenges to achieving universal interoperability. Today's complex distributed systems are characterized by extreme heterogeneity and spontaneous interactions. Computational devices ranging from embedded devices, sensors, and smartphones through to cluster machines and the Cloud use a wide range of communication networks and middleware protocols to communicate with one another. However, as soon as two systems adhere to heterogeneous protocols (from application down network layers) to interact with each other, interoperability is impossible. Standards are a well-established approach to rectifying these types of problems, eg the set of CORBA standards from the OMG, and the set of Web Services standards from the W3C. Where two systems agree upon such a standard, interoperability can be guaranteed. However, systems may encounter one another spontaneously where no such agreement is possible, and hence where the communication protocols differ they cannot interoperate.

The CONNECT project, which began in February 2009 and concludes at the end of 2012, is an EU Future and Emerging Technologies project involving partners whose expertise covers middleware, software engineering, formal methods, machine learning, software synthesis and systems dependability. The aim of the project is to overcome interaction protocol heterogeneity at all layers, on-the-fly, by using a revolutionary approach that dynamically generates the necessary interoperability solution to connect two heterogeneous systems. We term this new style of middleware: Emergent middleware.

Figure 1 illustrates an emergent middleware solution. The figure shows a CONNECTor ensuring interoperation between two networked systems and performs two important functions:

1. *Message interoperability* is dedicated to the interpretation of messages from/toward networked systems (listeners parse messages and actuators compose messages); by plugging in the correct listeners and actuators we can communicate with any legacy protocol.
2. *Behavioural interoperability* mediates the interaction protocols run by the communicating networked systems by translating messages from one protocol to the other.

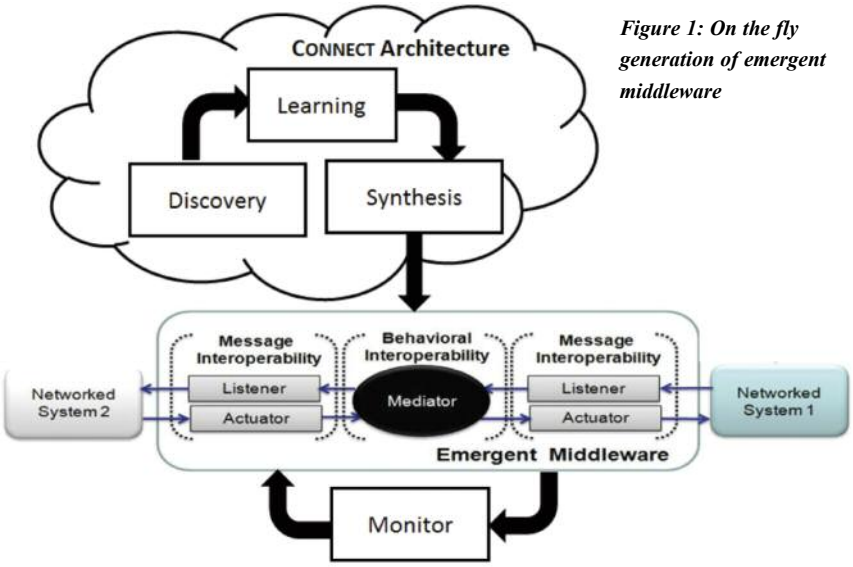


Figure 1: On the fly generation of emergent middleware

In order to dynamically generate the above solution on-the-fly, CONNECT employs a runtime architecture that executes the following phases (these phases are depicted in Figure 1):

1. *Discovery. Systems* in the local environment are discovered and their information (eg an interface description) is used to build a model of the service provided and what middleware it utilizes. Ontological information is then used to understand this provision and obtains better matching with other systems, ie does one system provide the behaviour another requires (irrespective of the heterogeneity)?
2. *Learning*. Active learning algorithms determine more precisely the behaviour of a given system in terms of the sequence of operations executed to provide a particular service and therefore enhance the model of a system.
3. *Synthesis*. The models of the two systems are used to calculate the necessary mediator that will ensure the translation between two systems. The generated mediator is used to create an emergent middleware, aka CONNECTor, which is deployed to achieve the interoperability between two systems.
4. *Evolution*. The deployed CONNECTor is monitored to ensure it is meeting the non-functional requirements of the connection (eg performance in terms of a minimum level of network latency). Where the requirement is not met, the architecture cycle is re-run to generate a new CONNECTor.

The above functionalities are supported by dedicated Enablers that are being developed as part of CONNECT and

released open source from the project Web site. In particular, the Starlink open source software framework offers the tools to develop Message interoperability solutions. Currently, Starlink has been successfully used to dynamically generate CONNECTors directly between middleware protocols such as: CORBA to XML-RPC, and Service Location Protocol to Bonjour; it has also been used to build bridges between heterogeneous application systems eg a Flickr client application built using XML-RPC interoperating with the Picasa REST API.

CONNECT has illustrated the possibilities of emergent solutions. However, the inherent openness and probability of such solutions raise important challenges, especially when deployed at Internet scale. Here it must be reliably able to produce correct mediators and also be secure against malicious threats, which are other areas of active research in CONNECT.

Links:
http://www.connect-forever.eu
Starlink framework:
http://starlink.sourceforge.net

Please contact:
Paul Grace
Lancaster University, Lancaster, UK
E-mail: p.grace@lancaster.ac.uk

Gordon Blair
Lancaster University, Lancaster, UK
E-mail: gordon@comp.lancs.ac.uk

Valerie Issarny
Inria, France
Arles project-team
E-mail: Valerie.Issarny@inria.fr

Never-stop Learning: Continuous Validation of Learned Models for Evolving Systems through Monitoring

by Antonia Bertolino, Antonello Calabrò, Maik Merten and Bernhard Steffen

Interoperability among the multitude of heterogeneous and evolving networked systems made available as black boxes remains a tough challenge. Learning technology is increasingly employed to extract behavioural models that form the basis for systems of systems integration. However, as networked systems evolve, their learned models need to evolve as well. This can be achieved by collecting actual interactions via monitoring and using these observations to continuously refine the learned behavioural models and, in turn, the overall system. This approach is part of the overall CONNECT approach.

In today's networked world, software is highly pervasive and increasingly existing components and services are dynamically aggregated into newer composite systems according to ever changing and demanding user expectations. The process of system integration and validation would presuppose the availability of models for the composition of software pieces. However, in real life software components that are released as black boxes do not generally provide models; this is particularly true with respect to behaviour. Learning techniques are thus being leveraged to enable the automatic derivation of formal behavioural models for black box components.

In active automata learning (see Figure 1), queries composed of a target system's input symbols are launched for execution. Based on the system responses, the learning algorithm creates and incrementally refines a hypothesis model of the system's behaviour.

However, this hypothesis is not guaranteed to be correct, which of course has implications for the correctness of the integrated system. Moreover, even if the constructed model was correct when it was built, the systems learned might subsequently evolve making the corresponding automata obsolete.

To tackle such issues, we propose a Never-stop Learning approach, where learning is not considered complete with

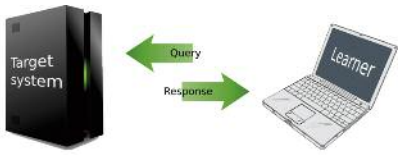


Figure 1: Active learning

the building of the model. Instead, we keep the black-box components under observation by run-time monitoring in order to capture the event flow of components' interactions within the integrated system, so that we can check whether the actual observed behaviour complies with the hypothesized model. We then attempt to retrace the sequence of monitored observations on the states of the current learned model. If this is not possible, ie there are no states in the hypothesis that can explain the observed interactions, this provides evidence of a mismatch between the learned hypothesis and the actual target systems.

The sequence of real-life observations that are not contained in the learned model can then be transformed into a counterexample for the learning algorithm, by retracing the observed steps in the hypothesis until diverging behaviour becomes apparent. This counterexample will thus trigger the construction of a refined hypothesis by the learning algorithm.

In Figure 2 the overall learning and monitoring approach is illustrated: The (publish/subscribe) monitoring infrastructure collects (through the instrumented connector) and makes sense of relevant data about the target system behaviour. The grouped observations are retrieved from the bus by the state tracker and matched against the state space of the learner's current hypothesis. If no such matching is possible, the trace of incoming observations is evidence of model mismatch,

which then is transformed into a counterexample by the evidence evaluator. Counter-examples are provided to the learning algorithm to update the model accordingly.

The approach is currently undergoing implementation within the European Future and Emerging Technologies project CONNECT, which addresses cross-platform, cross-middleware and cross-paradigm interoperability barriers. CONNECT envisions a dynamic environment populated by heterogeneous networked systems willing to communicate. The project aims at seamlessly supporting their interaction via mediating software bridges, called CONNECTors, which are synthesized and deployed on-the-fly.

The generation of these CONNECTors relies on learning technology to extract behavioural models for the networked systems which are fed to the synthesis enabler. In CONNECT, the learning is done via LearnLib, a flexible component-based framework for active automata learning, and monitoring is performed by GLIMPSE, a publish-subscribe event-based monitoring infrastructure. The synthesized CONNECTor is equipped with monitoring probes, which collect information on actual run-time system interaction. GLIMPSE then makes sense of the collected data, employing a complex event processor engine and rule-based delegation of observations. Transported over a shared message bus, the elaborated information is retrieved by the learning component in the CONNECT architecture and subsequently analysed.

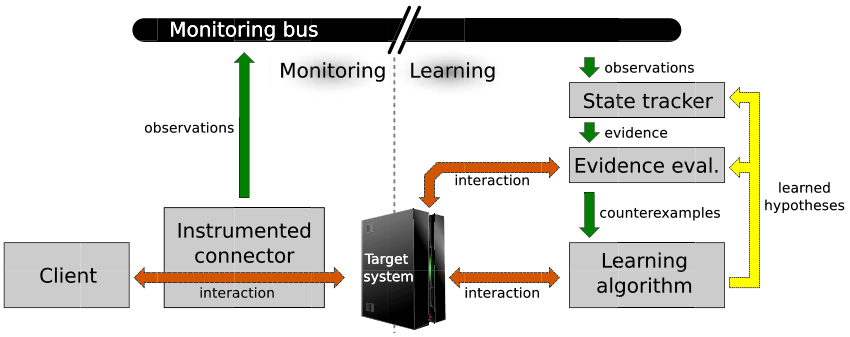


Figure 2: Never-stop Learning approach: learning and monitoring combined.

The described coupling of learning and run-time monitoring is just one example where technology is combined in novel ways to realize the CONNECT ambitious vision of eternal connection. The project, currently in its third year, will release a complex architecture populated with enablers not only for learning and monitoring, but also for discovery, synthesis, trust management, dependability and performance analysis, which altogether transparently will support interoperability among heterogeneous and evolving networked systems.

Links:
http://www.connect-forever.eu
http://labsewiki.isti.cnr.it/labse/tools/glimpse/public/main
http://www.learnlib.de

Please contact:
Antonello Calabrò
ISTI-CNR, Italy
Tel: +39 050 315 3463
E-mail: antonello.calabro@isti.cnr.it

Maik Merten
Technische Universität Dortmund, Germany
Tel: +49 231 755 7759
E-mail: maik.merten@tu-dortmund.de

PINCETTE – Validating Changes and Upgrades in Networked Software

by Pamela Farries and Ajitha Rajan

Europe relies on the availability and flawless functioning of distributed infrastructure services such as electricity, water, communication, transportation and environmental management, which are all increasingly controlled by software. Currently, however, the potential for innovation is limited owing to the inherent risks and costs of upgrades. For example, incompatibilities when simultaneously running old and new versions of control software can result in major service outages. Similarly, fast moving technology, such as sensors, could enable advances in safety critical applications such as aerospace; however revalidating avionics software to introduce new niche functionality is prohibitively expensive. The PINCETTE project has been set up specifically to ensure safe infrastructure upgrades and enable rapid product development by providing solutions for continuous validation. Certainty in the safety of upgrades will allow communities to confidently apply and leverage the efficiency gains and advantages of upgrades.

The PINCETTE project targets the problem of analyzing and validating complex systems upgrades. London's power grid, for instance, has over 4000 medium-voltage substations; the software in controllers for these substations must prevent power outages and ensure safety. A malfunctioning device can result in huge economic damage and can even threaten human lives. Upgrades to software might be needed to improve efficiency or to fix a bug, but validating this upgrade is complicated owing to the software being distributed across substations and the entire grid. Additionally, the grid cannot be switched off when validating an upgrade, making upgrades across such a network a huge safety concern.

With current technology, the only option is to supplement each incremental

Photo: National Grid

The Electricity National Control Centre for Great Britain

change or upgrade in the system with a complete revalidation of the whole system, incurring enormous cost in the process. The need to conduct total system checks arises from the fact that state-of-the-art testing and formal verification tools are not optimized to validate system changes and upgrades, but instead focus on a single program version only. For instance, in a new release of software or an operating system, bugs are often found post release and patches are then distributed. However, this approach which relies on the discovery of bugs created by upgrades during use of the product and subsequently fixing them is not acceptable in many domains, particularly safety critical systems such as aerospace, medical devices, nuclear reactors etc, when it is crucial to ensure the absence of bugs for every upgrade. The associated cost can lead to a very conservative attitude to changes and a failure to fully exploit potentially beneficial advances in technology.

In the PINCETTE project, we are working towards resolving this problem by developing the technology to ensure safe infrastructure upgrades by validating continuously evolving networked software systems. Our goals are to 1) reduce the cost and time to market of upgrades by several orders of magnitude; 2) increase the level of confidence in the safety of upgrades; 3) enable certification of upgrades. The notion of 'system upgrade' in PINCETTE is broad and includes changes of functionality, bug fixes, feature upgrades, and requirement changes.

In the PINCETTE approach for upgrade validation, we do not revalidate the new version of the system in its entirety. Instead we only verify the safety of the portion of the system affected by the upgrade and the impact of the upgrade on the new version. This is done by utilising the knowledge that the behavior of the existing version of the system is safe. Doing this will drastically reduce the time and effort required to revalidate the new version and result in dramatic cost savings. To achieve this, we are developing technologies to assess the impact of changes. We plan to use a combination of state-of-the-art static and dynamic analysis techniques for efficient analysis of system changes. Dr Hana Chockler's team at IBM Haifa, Prof Daniel Kroening's group at the University of Oxford, Prof Natasha Sharygina's group at Universita Della Svizzera Italiana (USI) and Dr Leonardo Mariani's group at Universita' Degli Studi Di Milano-Bicocca (UniMiB) are working together to develop the technology for change impact analysis and verification. The tools developed will be tested and validated on real world systems:

- A UAV (Unmanned Air Vehicle) observation payload example provided by IAI (Israeli Aerospace Industries)
- Software for high-voltage and medium-voltage substation automation systems offered by ABB
- A control system for remote maintenance operations of the ITER fusion reactor provided by VTT (The Technical Research Centre of Finland).

The project commenced in July 2010 and will run for three years under funding from FP7 ICT-2009.1.4 Trustworthy ICT.

Links:
http://www.pincette-project.eu/
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5770962

Please contact:
Pamela Farries
Department of Computer Science,
University of Oxford
Tel: +44 1865 610736
E-mail: pamela.farries@cs.ox.ac.uk

Automatic Upgrade of Java Libraries

by Zdeněk Troníček

Our work focuses on the automatic update of Java code in response to changes in library interfaces. Our approach is based on refactoring rules that are specified by the library author and later applied by a developer on client code. This approach turns out to be very successful in practice. Although many researchers have focused on the problem of automatic migration to a new API and several approaches to the problem have been suggested, there is currently no widely used tool for automatic migration and research is still active in this area.

The Java programming language is one of the world's most popular programming languages. It was introduced in 1996 as a simple programming language based on C syntax and it has been evolving ever since. Over the last 15 years, it has evolved to a mature programming language with advanced features such as parametric polymorphism.

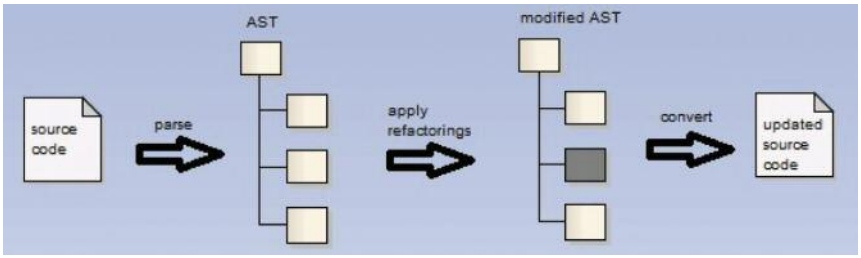


Figure 1: Workflow in RefactoringNG and JUpgrade

From the beginning, the Java programming language has had standard library classes (Java API) that have been evolving along with the language. In the course of the evolution many new classes and methods have been introduced and many have been deprecated. For example, the enable() method in the java.awt.Component class was deprecated and replaced by setEnable(boolean b) in Java 1.1. This approach to API evolution is motivated by backward compatibility: A source code that compiled in Java 1.0 should compile in Java 1.1 as well. The same approach to API evolution is commonly used in Java libraries. For example, Simple API for XML Processing (SAX) version 2 deprecated six classes from SAX version 1 and introduced new classes to replace them. Although this is profitable for source compatibility, it has also two significant disadvantages: (a) API developers have to maintain several APIs in parallel and (b) deprecated methods inhibit API evolution because API developers are restricted by them.

In many cases, deprecated and new classes and methods are similar because the change in API was caused by some kind of refactoring, ie a structural transformation that does not change code behaviour. The fact that many API changes are caused by refactoring provides a good incentive for the development of tools that are able to apply the same refactorings on client code. Such tools can facilitate migration to a new version of the library. The reasons for migration may vary and might include, for example, bug fixes or performance improvements. Upgrade to a new version of the library involves changes in source code that are currently usually done manually. This is boring, tedious and error-prone.

Our research is focused on tools that are able to adapt Java source code to a new library interface provided that the changes in the library interface were caused by refactorings. Such tools can reduce the workload for programmers. In addition, they can facilitate maintenance of API because deprecated classes and methods may be safely removed if clients can migrate to new classes and methods automatically.

We designed two tools, RefactoringNG and JUpgrade, that can apply specified transformations on Java source code. As well as many other transformation tools, they do not work directly with source code but apply transformations on the abstract syntax trees (ASTs). The ASTs are built and attributed by the Java compiler and we access them through the Compiler Tree API. Transformations are applied in three steps: first, we let the compiler build and attribute the ASTs; second, we apply transformations (this typically modifies some ASTs); third, we convert the modified ASTs to source code. Transformations that can be applied vary between RefactoringNG and JUpgrade. RefactoringNG can only rewrite ASTs to other ASTs. Each AST transformation here is described by a rule that consists of two ASTs: the pattern tree and the rewrite tree. The tool searches for the pattern tree and when found, it rewrites the pattern tree to the rewrite tree. In comparison to other tools, RefactoringNG has available complete syntactic and semantic information.

JUpgrade uses a higher-level approach. We specify refactorings we want to apply to source code and the tool applies them without human intervention. These refactorings can be, for example, "rename method" or "change method signature". Before applying a refactoring, the tool verifies that the refactoring is valid in client context. For example, "rename method" can be applied only if no method with given signature exists in target class. As for API clients, we distinguish between instantiators and extenders. The instantiator instantiates classes and uses classes, interfaces, enums, and annotations. Thus it sees only public members of types. The extender extends classes and extends and implements interfaces. Besides the public members, it sees protected members as well. When migrating from SAX 1 to SAX 2 with JUpgrade, we successfully migrated 43 out of 50 method calls in complete instantiator and 46 out of 50 methods in complete extender.

Link:
http://kenai.com/projects/refactoringng

Please contact:
Zdeněk Troníček
Faculty of Information Technology,
Czech Technical University in Prague
E-mail: tronicek@fit.cvut.cz

A Benchmark for Design Pattern Detection Tools: a Community Driven Approach

by Francesca Arcelli, Andrea Caracciolo, Marco Zanoni

Comparing tools is a difficult task because standard benchmarks are usually not available. We propose a Web application that supports a collaborative environment for the comparative analysis of design pattern detection tools.

Design pattern detection (DPD) is a topic that has received much attention in recent years. Finding design pattern (DP) instances in a software system can give useful hints for the comprehension and evolution of the system and can provide insight into the design rationale adopted during its development. Identifying design patterns is also important during the re-documentation process, in particular when the documentation is poor, incomplete, or out of date.

Several DPD approaches and tools have been developed, exploiting different techniques for detection, such as fuzzy logic, constraints solving techniques, theorem provers, template matching methods, and classification techniques. However, the results obtained by these tools are often quite unsatisfactory and differ from one tool to another. Many tools find pattern candidates which are false positives, missing other correct ones. One common difficulty in DPD is the so-called variant problem: DPs can be implemented in several different ways. These variants are the cause of the failure of most pattern instance recognition tools using rigid detection approaches, based only on canonical pattern definitions.

In fact, it is difficult to perform a comparative analysis of design pattern detection tools and only a few benchmark proposals for their evaluation can be found in the literature. In order to establish which are the best approaches and tools, it is important to agree on standard definitions of design patterns and to be able to compare results. The adoption by the DPD community of a standard benchmark would improve cooperation among the researchers and promote the reuse of existing tools instead of the development of new ones.

With this goal in mind, we have developed a benchmark web application to compare the results of DPD tools (see Figure 1). Subsequently we aim to extend and refine this application through the involvement of the community.

Our approach is characterized by:

- a general meta-model for design pattern representation;
- a new way to compare the performance of different tools;
- the ability to analyze instances and validate their correctness.

Our aim is not only to introduce a "competition" between tools but also to create a container for design pattern instances that, through user testing and evaluation, will lead to the building of a large and "community validated" dataset for tool testing and benchmarking.

We are interested in integrating other comparison algorithms, preferably after discussions with and input from the DPD community, in order to let users choose the algorithm they think most appropriate. We are also investigating the possibility of installing some kind of web service that allows registered users to automatically upload their analysis into the application.

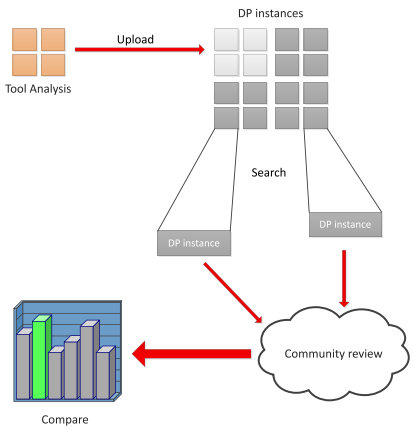


Figure 1: A schema of the benchmark definition process. Users upload tool analyses, contribuiting to the build up of a repository of DP instances. Each instance can be searched, analyzed and reviewed. The resulting data are used to create a benchmark of the associated tools by comparing their estimated precision rates.

We have included in our benchmark application the contents of PMARt , a repository developed by Y. G. Guéhéneuc, that contains the pattern identification analysis of specific versions of different open source programs, in order to provide a good reference for comparing results and identifying correct instances. In addition, we have already uploaded some results provided by the following DPD tools: WoP, developed by J. Dietrich and C. Elgar, and DPD-Tool, a tool developed by N. Tsantalis, A. Chatzigeorgiou, G. Stephanides, and S. T. Halkidi.

We are interested in testing the correctness and completeness of our approach by exchanging data with other applications and models, such as DEEBEE (DEsign pattern Evaluation BEnchmark Environment), a web application for evaluating and comparing design pattern detection tools developed by L. Fulop, R. Ferenc, and T. Gyimothy.

The availability and simple interchange of DPD results could also be helpful to support DPD techniques such as those proposed by G. Kniesel and A. Binum, who compare different design pattern detection tools and propose a novel approach based on a synergy of proven techniques.

Through our online web application, we intend to provide a flexible underlying model and aim at supporting the comparison of DPD results in a collaborative environment.

Link:
http://essere.disco.unimib.it/DPB/

Please contact:
Andrea Caracciolo
University of Milan-Bicocca, Italy
E-mail: a.caracciolo1@campus.unimib.it

Code Smells, Micro Patterns and their Relations

by Francesca Arcelli Fontana, Marco Zanoni and Bartosz Walter and Paweł Martenka

Code smells are an example of a pattern-oriented trend applied to software evolution. Due to their abstract and informal nature, the detection process requires support from lower-level indicators like individual metric values, history of code changes, knowledge about program structure or results of program execution. We examine how the presence of program structures, called micro patterns, is correlated with the code smells and how micro patterns can be exploited for the purpose of smell detection.

Over the last 15 years pattern-oriented research has been a growing field in software engineering. Starting with design patterns, the idea of providing generic, proven solutions to problems of a diverse nature has reached nearly every stage of software development cycle. Patterns are finally being applied to software evolution, which is currently one of the most important challenges in modern software engineering.

Code smells are high-level, intuitive and informal characteristics of program source code that can make software hard to maintain. Using a medical metaphor, they play the role of easily observable symptoms that may indicate a serious illness, but they do not provide information about the root cause. The original catalogue of code smells was proposed by Fowler in 1999 and has been successively extended.

Due to their vague and abstract nature, there is no a single detection method for a given code smell. A smell named Large Class, for example, describes a class that carries too much responsibility and probably should be split up. There are multiple indicators that can suggest the presence of a smell: numerous class members, many implemented interfaces or high internal complexity or low values of cohesion metrics. Therefore, a number of smell indicators should usually be analyzed. Previously we identified a number of such indicators and provided a few examples of how they leverage the detection of particular smells. They include for instance, metrics values, results of dynamic analysis, history of code revisions and knowledge about other flaws that have already been detected or rejected.

Micro patterns are another example of a pattern-related trend in software engineering. They are intended to capture common low-level programming techniques, both positive and negative. They can be thought of as class-level traceable patterns, ie structures similar to design patterns that can be mechanically recognized. The detection of micro patterns can provide hints on good and bad programming practices. In our research we analyze how micro patterns can be used as indicators of a code smell's presence.

Research plan

The research involves the following steps:

1. Identify hypothetical relationships that exist between micro patterns and code smells.
2. Conduct an experiment to verify the hypotheses formulated in step one.
3. Perform post mortem analysis to find associations between smells and micro patterns and their impact on the detection process.

Brief description of experiment

A number of smell-detection tools are currently on the market. Based on our experience and early experiments we chose two: InFusion and PMD, both designed for analyzing Java programs, but employing different definitions of detected smells.

The codebase for the experiment included six instances of software systems, coming from two open-source projects: GanttProject (in five versions) and JHotDraw (v. 7.0.9), with sizes varying from 393 to 800 classes.

In order to determine a number of smells in all six instances an initial analysis was performed. GanttProject instances had 11 different code smells which recurred – albeit with varying strengths - across almost all versions of the software. JHotDraw was found to be biased with 13 smells, in most cases reduntant with those detected in GanttProject. Similarly, all instances were also examined for the presence of micro patterns. We found examples of 13 different micro patterns in all instances of analyzed projects.

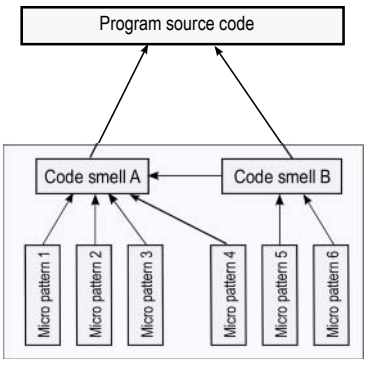


Figure 1: Hypothetical relationships between code smells and micropatterns

Analysis of results

In order to find relationships between micro patterns and code smells we first counted positive and negative matches of a given micro pattern to a particular smell, and calculated how they correlate. Next, the datasheet with smells and micro patterns has been processed with a Weka implementation of Predictive a priori algorithm. This step resulted in a set of rules that express the identified relationships between code smells and micro patterns, with a certain confidence value. It revealed 12 correlations with a confidence level higher than 50% (eg co-existence of Data Manager and Extender micro patterns implies the presence of Significant Duplication smell with accuracy of 71.5%). However, not all analyzed micro patterns appeared correlated with code smells. Only six (out of 13) patterns and three (out of 11) smells were included in the generated rules.

Conclusions

Our results indicate that the presence of some micro patterns is correlated with the existence of code smells, and can be used to support the smell detection process, and later to evaluate the internal quality of the software. However, the experiment involved a relatively small code base, so a more thorough investigation is needed.

Please contact:
Francesca Arcelli, University of Milano Bicocca, Italy,
E-mail: arcelli@disco.unimib.it

Bartosz Walter, Poznań University of Technology, Poland
E-mail: Bartosz.Walter@cs.put.poznan.pl

Pat-Evol: Pattern-Driven Reuse in Architecture-Based Evolution for Service Software

by Aakash Ahmad and Claus Pahl

Although architecture-centric maintenance and evolution is useful for adjusting software structure and behaviour at higher abstractions, this approach lacks the potential for systematic change reuse. The Pat-Evol project focuses on managing a constructive architecture-based evolution process for service software. It enables continuous automated identification of evolution patterns from the architecture change log with support for pattern specification and instantiation provided by a pattern library. A pattern library acts as a repository to enable pattern-driven change execution, supporting the notion of an off-the-shelf architecture evolution.

Service Oriented Architecture (SOA) is a business-centric architectural approach representing business processes as technical software services. In this context, architecture-centric maintenance and evolution can be exploited to adapt service process structure and behaviour at higher levels of abstraction. The emergence of change patterns or evolution styles promotes the build-once use-often philosophy in component-based architectural evolution, but often falls short of addressing frequent, demand-driven process-centric change in SOAs. We focus on exploiting recurrent evolution patterns that are specified once and instantiated multiple times thus providing a generic, reusable solution to recurrent SOA evolution problems.

The Pat-Evol Project

Pat-Evol - an SFI funded project at Lero, the Irish Software Engineering Research Centre – focuses on enabling pattern-driven reuse in architecture evolution for service software. Within Lero competency areas, the project aims at providing a formal foundation and tool support for the evolution of business-critical software. It enables an automated identification of evolution patterns with repository support for pattern specification and instantiation during evolution. The novelty in the project lies in:

- Exploiting the architecture change log (a history of sequential change) to continuously identify evolution patterns that provide a generic solution to recurring architecture evolution problems.
- Support for pattern specification and instantiation through a pattern library that comprises of a continuously validated and updated collection of patterns as reusable solutions to architecture evolution problems.
- An evolution application framework to enable pattern-based reuse during change execution to support the

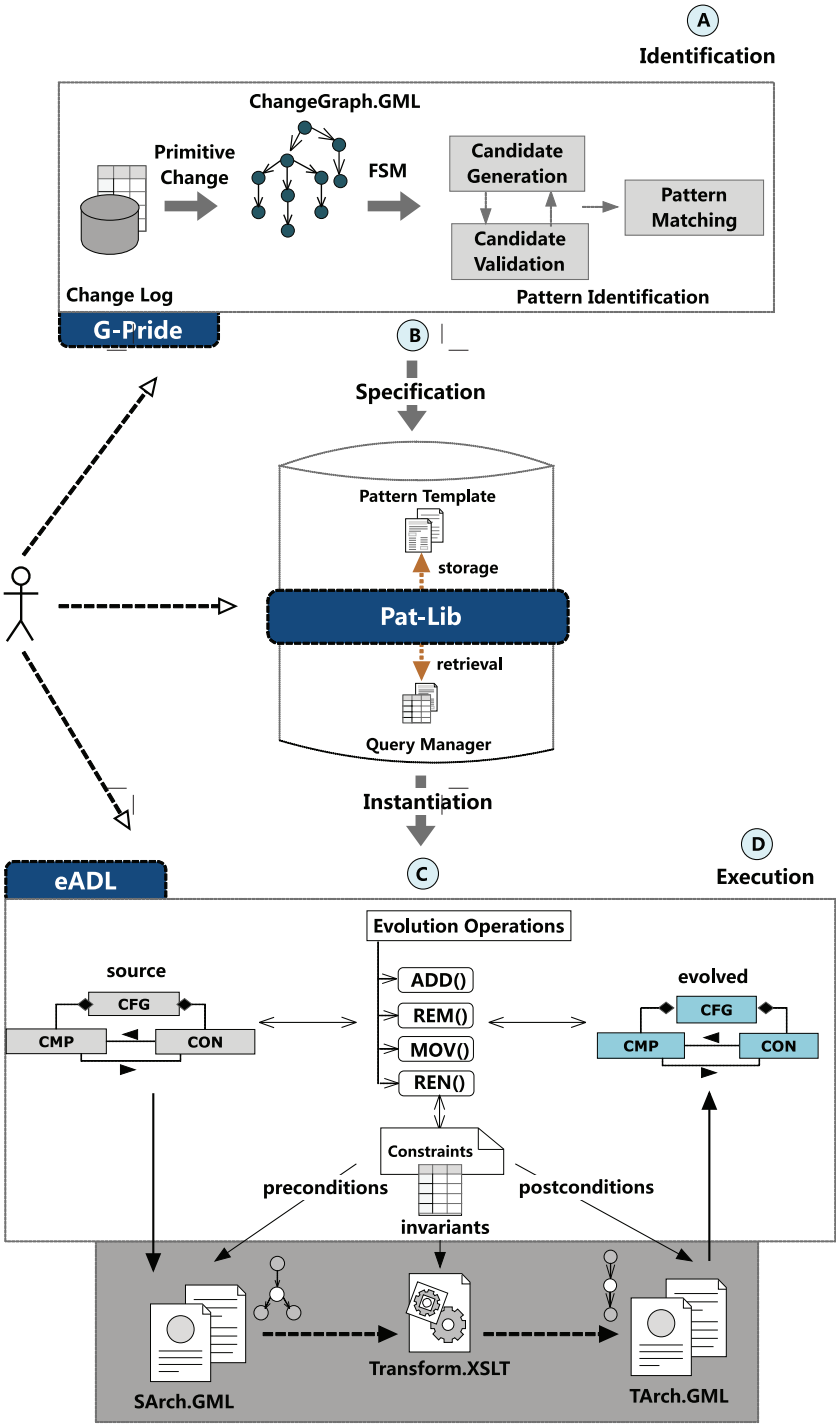


Figure 1: Reusable change execution for service architecture evolution

notion of off-the-shelf evolution in service architectures.

At the core of the Pat-Evol project is an evolution pattern identification technique that continuously detects and updates patterns and allows them to be utilized from a library.

Evolution Pattern Identification

Pattern identification aims at capturing the recurring primitive architectural changes that occur with change operations (add, remove, modify etc.) on architecture elements (configurations, components, connectors etc.). In order to achieve this, fine-grained instances of sequential architectural changes are retrieved from the architecture change log (ACL). The changes are modelled as a typed attributed graph providing formal semantics with its node and edge attribution to capture change operations on architectural elements. Sub-graph mining in the architecture change graph identifies exact instances of evolution patterns, but also inexact matches where only central pattern features suffice for identification.

In order to facilitate automation, the developed algorithm - G-Pride (Graph-based Pattern Identification) - uses an a priori-based approach to (i) generate and validate the pattern candidates and (ii) determine the occurrence frequency of candidates in an architecture change graph.

The Pattern Library

The library is a collection of evolution patterns which provide empirically determined generic and potentially reusable pattern-based solutions to a collection of architectural evolution problems. The primary benefit lies with i) automated population - to eliminate time-consuming and error-prone manual efforts for continuously validating and updating the pattern collection and ii) assisted retrieval - to assist the architects in selecting and instantiating appropriate pattern(s) for a given evolution context.

The Pat-Lib library prototype supports semi-automated specification of identified patterns using pattern templates. This allows architects the query-based retrieval of concrete pattern(s) instances during architecture evolution.

Reusable Change Execution

A framework for pattern-based reuse of change and evolution steps during architecture change management is based on constrained composition of evolution operations on architecture elements enabling their structural evolution. The architecture meta-model is represented as a typed attributed graph with its node and edge attribution capturing con figuration among components and connectors. This allows for modelling the evolution operations as (constrained) graph transformation rules to enable structural evolution of architectures.

We are currently developing the eADL (Architecture Evolution Description Language) that allows sequential composition of change to enable pattern-based change execution. The architecture (graph) model is expressed as an XML structure that allows for the execution of evolution operations on architecture elements as the XSLT rules.

Conclusions

Automating the change pattern detections allows us to tailor change management to particular environments and to validate and update these change patterns continuously. The Pat-Evol project aims to provide graph-based formalism and tool support to manage change reuse during structural evolution in service architectures. It provides automation and tool assistance for architects to obtain properly documented and up-to-date descriptions of architectures and their evolution based on common evolution patterns.

Links:
http://www.lero.ie/research
http://www.computing.dcu.ie/~cpahl/

Please contact:
Claus Pahl
Dublin City University, Ireland
Tel: +353 17005620
E-mail: Claus.Pahl@computing.dcu.ie

Architectural Evolution with Adaptive Services: the Meta-Service Approach

by Carlos E. Cuesta, M. Pilar Romay and Elena Navarro

At the architecture level, the application of adaptation strategies often implies dynamic reconfiguration and, ultimately, system evolution. Hence adaptation can be considered as a low-level version of architectural evolution; generic dynamic techniques, such as meta-level superimposition, can effectively be used for both. This approach can be translated to several different contexts, and we describe the specific case of service-oriented architectures.

The conceptual relationship between software evolution and adaptation is well known. Nonetheless, they have evolved into full-fledged research fields, acquiring new features and connotations; hence this relationship has to be carefully analyzed in different contexts.

In Software Architecture, the system is studied as a whole; and then the impact of evolution has to be considered system-wide. From the beginning, this has been a key concern in architectural research; indeed, the notion of architectural style was specifically conceived for the needs of evolution. Moreover, the area of dynamic architecture was designed to combine both issues, usually within the limits of a style. In recent times, the specific notion of evolution style has been conceived of to provide an analogous abstraction, within the area of architectural dynamism.

Meanwhile, self-adaptive architectures, those able to adapt to changes in the environment, stress the limits of existing dynamic techniques. Only the most general approaches are able to express the needs of self-adaptive com-

ponents. Among these, the most powerful are probably the autonomic control loop and reflection. In both cases, a part of the system, ie services, is devoted to perform system tasks, while another is in charge of managing these services. These meta-services were originally designed to control evolution; currently they also are in charge of performing required adaptations. In fact, as summarized in Figure 1, when an adaptation has to modify the system's structure and/or behaviour, it is actually executing a step of dynamic evolution.

Reflective architectures were introduced roughly a decade ago. They encompass two levels: the meta-level, which is able to operate on the system itself, and the base-level, which is adapted by these operations. Almost any potential change can be described using this approach; but the implicit capability to modify the meaning of internal concepts is often considered too complex.

An alternative perspective can be offered, using the notion of superimposition. Instead of having meta-components which manage the concepts of the language itself, this approach defines interactive supervisors which impose their control on selected components. Thus, they provide the same dynamic effects, just by observing and altering their interactions. Note also that this just defines a conceptual layer: the adaptive structure is still essentially decentralized, providing the basis not only for adaptation, but also for self-organization.

Several proposals have covered these issues from a generic perspective; in our own research we have developed the reflective architectural language PiLar, to describe dynamic architectures, and also an equivalent aspectual language, specifically designed on top of superimposition. Both are based on the π-calculus, which defines a sound formalization for the approach; besides, this choice happens to provide another coincidence with service technologies.

Self-adaptive approaches are relatively new, and their application to architecture is even more recent. In particular, the way to provide self-adaptive properties in specific architectural styles can differ depending on the application

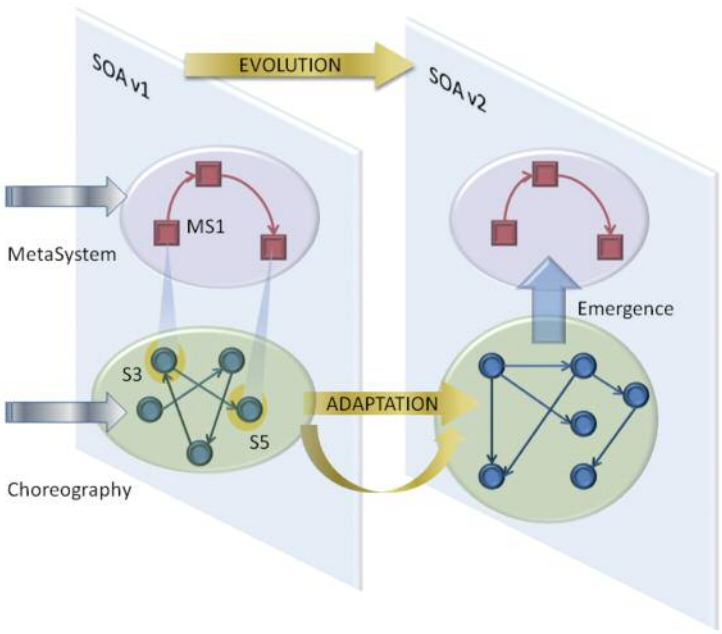


Figure 1: Decentralized adaptation of services may imply system-level architectural evolution.

domain. We have developed both a theoretical foundation for adaptive architectures, and the application of this theory to multi-agent systems and agreement technologies. Even considering their commonalities, these versions are clearly distinct from each other.

Furthermore, the service-oriented domain is particularly interesting; services have specific features which separate them from conventional components. In particular, a service has a unique locator and can be accessed even when it is composed. Therefore, it can perform a dynamic adaptation even from inside a coordination process, whether a strict orchestration, or a more generic choreography.

The relationship between a meta-service and its managed services is therefore much more flexible than other constructions. Figure 1 depicts a number of services within a SOA, which are affected by the corresponding meta-services, inducting an adaptation in their behaviour. This adaptation may or may not alter the choreography. The service itself will be able to decide about this, and hence it becomes an adaptive service.

As already noted, the adaptation can modify the structure of the service architecture, hence implying an evolution in the system. But the corresponding scenario can be even more complex: sometimes the resulting structure has emergent properties, which must be distilled and considered at the meta-level. Therefore, in such a case our system-wide evolution happens effectively at two different levels of abstraction.

Our current work is exploring the properties and implications of this kind of adaptive service architecture, as well as its relationship with the notion of evolution style, even in the context of product lines; and also considering co-evolutionary transformations of the design decision model.

Links:
VorTIC3 Research Group (Rey Juan Carlos University):
http://www.vortic3.com

Agreement Technologies (CONSOLIDER Project):
http://www.agreement-technologies.org

Please contact:
Carlos E. Cuesta, Rey Juan Carlos University, Madrid, Spain
E-mail: carlos.cuesta@urjc.es

M. Pilar Romay, St. Paul-CEU University, Madrid, Spain
E-mail: pilar.romay@gmail.com

Elena Navarro, Castilla-La Mancha University, Albacete, Spain
E-mail: Elena.Navarro@uclm.es

Cross Modality Adaptation of Service Front Ends

by Fabio Paternò, Christian Sisti and Lucio Davide Spano

The current evolution of pervasive technologies means that traditional access modalities are not always suitable for a particular service and, consequently, need to be reassessed. There are many scenarios in which the user is unable to use a screen or keyboard to interact with a given service. In such cases, a vocal interaction can be useful, but this is only supported by a few application providers. We propose a solution to adapt existing Service Front Ends in order to support vocal interaction, which is based on reverse engineering, adaptation heuristics and automatic generation of the vocal implementation.

Pervasive environments imply the possibility of accessing interactive applications anywhere and anytime. There are times, however, when a user may need to access a service but is involved in tasks that occupy his or her hands, making it impossible to use a screen or keyboard. The vocal modality is a good candidate for service access in these situations, and this is confirmed by the recent effort dedicated to supporting it by the main players in mobile OS (eg iOS and Android).

Though supporting vocal interaction can be a good opportunity for service providers, the creation of a Service Front End (SFE) for an additional modality requires an effort that not all application developers can afford. Our approach addresses this issue by supporting the automatic adaptation of existing Web SFEs to the vocal modality.

Our solution is based on an adaptation server, which is able to produce a VoiceXML SFE taking as input its HTML+CSS implementation. The final result adapts the hierarchical information structure of the initial SFE, by including menus that facilitate vocal navigation of the content and remove elements not relevant for the vocal rendering. Both input and output techniques are also adapted so that they can be supported by the vocal modality. In this process we exploit the MARIA language, which provides an abstract SFE language (with a vocabulary independent of the interaction modality) and a set of concrete SFE languages in which the abstract vocabulary is refined taking into account the target interaction platform.

The adaptation process consists of three main steps, implemented by the following modules:

1. The Reverser parses the HTML and CSS implementation, and creates a MARIA model-based, logical description of the SFE, including contents and semantic information such as groupings, heading, navigation etc. (some are already included in the HTML, others are obtained by applying specific heuristics). This description is tied to a graphical desktop vocabulary.

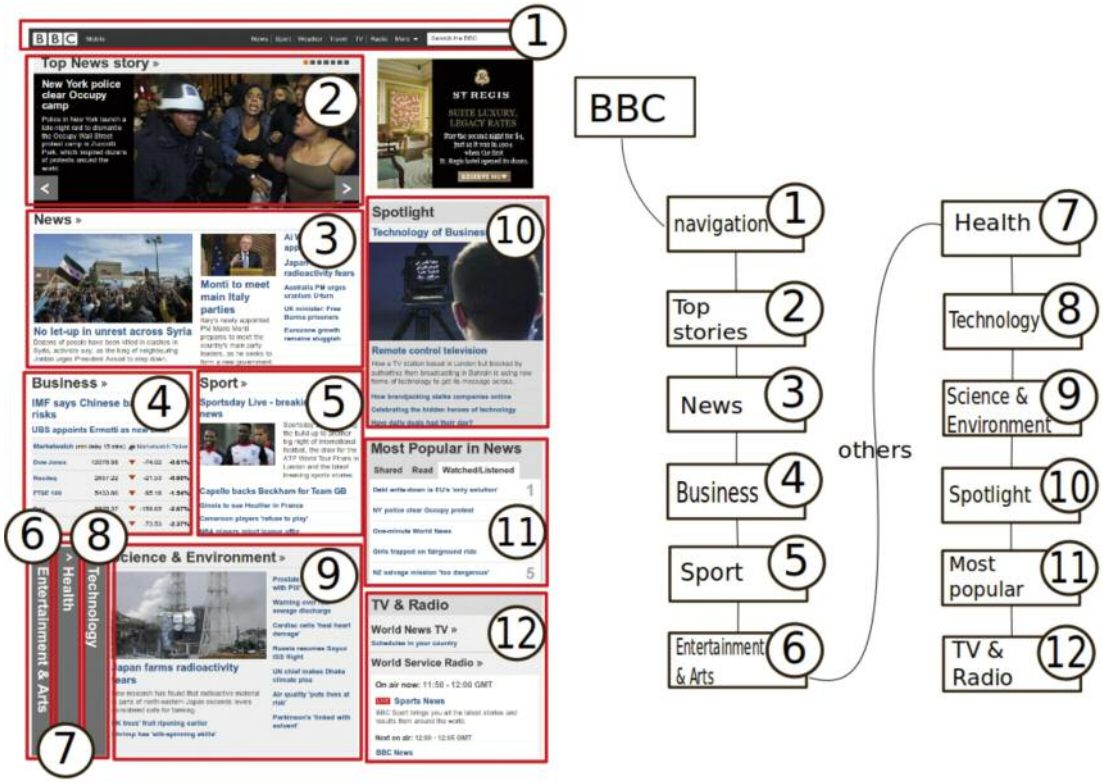


Figure 1: Cross Modality adaptation example

2. The Graphical-to-Vocal Adapter transforms the Reverser output into another MARIA model-based, logical description, switching the vocabulary from graphical desktop to vocal. This step consists of three sub-steps. The first performs a content and structure optimization, removing elements that are only used for layout purposes and splitting the contents so that they can be easily browsed vocally. The second step creates the vocal menus that support the interface navigation, finding meaningful labels for the identified content groups. The last transforms the desktop elements into vocal ones (ie a drop-down list to a vocal choice among a predefined set of options, images to alternative text descriptions, submit buttons to confirmation prompts, etc.).
3. The Voice XML Generator takes as input a MARIA SFE description for the vocal platform and translates it into a Voice XML application that can be accessed through vocal browsers.

Since the adaptation is performed on the logical SFE descriptions it is independent of the implementation languages. In addition, the adaptation process can be customized by SFE designers through a tool that allows a set of parameters to be changed (eg the maximum number of items in a vocal menu). The menu structure that will be generated can also be previewed and updated when a parameter has been changed.

Figure 1 shows an example of the desktop-to-vocal SFE adaptation on the BBC News site. The left part represents the original page layout. The adaptation process identified a set of different groups, which are highlighted with the red boxes. It then created the navigation structure, shown in the right part of the figure. The text within the boxes shows the words the user has to pronounce in order to move to the corresponding section. Finally, each page element is transformed into its vocal counterpart.

The resulting application will start asking the user to choose between the first set of parts in the BBC homepage (Navigation, Top stories, News, Business, Sport, Entertainment & Arts) or others. If, for instance, the user says "others" then the remaining parts of the page will be listed (Health, Technology etc). If the user says "News", the application will start reading the latest news included on the BBC website. The user is always able to go back to the previous menu, restart the navigation or exit the application by pronouncing the words "Back", "Previous" or "Exit", respectively.

We have presented a solution to make Web pages more suitable for vocal browsing by analyzing and modifying their logical structure. A customization tool was also developed in order to interactively change the parameters of the adaptation process and have a pr view of the structure of the generated vocal application.

Links:
HIIS Lab: http://giove.isti.cnr.it
MARIA home page and Environment:
http://giove.isti.cnr.it/tools/MARIA/home
SERENOA EU Project:
http://www.serenoa-fp7.eu/

Please contact:
Fabio Paternò
ISTI-CNR, Italy
E-mail: fabio.paterno@isti.cnr.it

CAPucine: Context-Aware Service-Oriented Product Line for Mobile Apps

by Carlos Parra, Clément Quinton and Laurence Duchien

The design of a mobile application is a tedious task owing to its intrinsic variability. Software designers must take into account in their development process the versatility of available platforms (eg Android, iPhone, tablets). The variety of existing devices and their divergences (eg frontal camera, GPS) introduce a further layer of complexity to the development process. In addition, at runtime, many potential situations have to be considered (eg limited connectivity, hardware heterogeneity, changes of user preference, etc.). Such software systems are seen increasingly as evolutive systems.

To tackle such a challenge, a cornerstone element that is intrinsic to any modern software is the notion of adaptation. If a system is designed and implemented to be adapted, then it is more likely that it can better support changes in its requirements, architecture, and even implementation. Software adaptations open the door for new categories of systems that use all the information available both at design time, to postpone and minimize the impact of developer decisions, and at runtime, to take advantage of the information available in the application environment. However, adaptations are difficult to define since they may take place at early stages of the development process, but also at runtime where there are many situations that have to be considered. Typically, this kind of information is designated as context information. Hence, a context-aware system is aware of changes in the context and is able to adapt to offer better user experiences. A well-known example of such behaviour is when a system changes its behaviour depending on the location. This is especially true in mobile computing.

In this project, we explore the applicability of the Software Product Line (SPL) paradigm to the development of adaptable systems for mobile applications. SPLs aim at managing and building multiple software products from a set of previously developed and tested assets. An asset is understood as any software artifact that can be

employed in the development of an application. In SPL engineering, commonalities and variabilities across a set of applications (ie product family) are identified, so that assets can be developed, and used to create different products.

The product derivation defines how assets are selected according to a given feature configuration, and specifies how those assets are composed in order to build the desired product. While the product derivation is commonly perceived as only belonging to the development process of products, this is not always the case and it can be extended to cover the adaptation of an existing product at runtime. In fact, the idea of using SPLs to derive dynamic products has recently started to gain interest from the academic community (see for example, the work of Hallsteinsen et al.). A Dynamic SPL (DSPL) is capable of producing systems that can be adapted at runtime in order to dynamically fit new requirements or resource changes.

Our DSPL, named CAPucine for Context-Aware Service-Oriented Product Line, proposes a unified approach that supports the complete software life cycle: from feature selection and initial product derivation, to runtime adaptation in response to changes of the execution environment. It is a complete approach for designing and implementing DSPL (see Figure). We concretize the notion of asset with a definition of aspect models to leverage the variability across a family of products. Derivation of products is divided into two adaptation processes: design time adaptation and runtime adaptation. The former is in charge of creating the initial product. The latter modifies the product once it has been created and deployed depending on the execution context. Finally our approach unifies design and runtime adaptations by representing both categories of adaptation as aspect models and brings benefits in terms of modularization, platform independence, reusability, and fast development.

As the first contribution, we introduce both: a simple - yet complete - variability model, and an aspect model that realizes variability. With the variability model we aim to define a family of products and identify commonalities and variabilities among them using variants. Additionally, the variability model allows the definition of constraints between those variants. The aspect model, on the other side, is used to construct platform independent representations of variability. Each aspect is self-contained in the sense that it has the three pieces of information required for it to be integrated into any product. It defines the model that represents the subsystem to be added, advice with a set of changes to the core application, and finally, pointcuts that identify the places where the aspect performs the modifications.

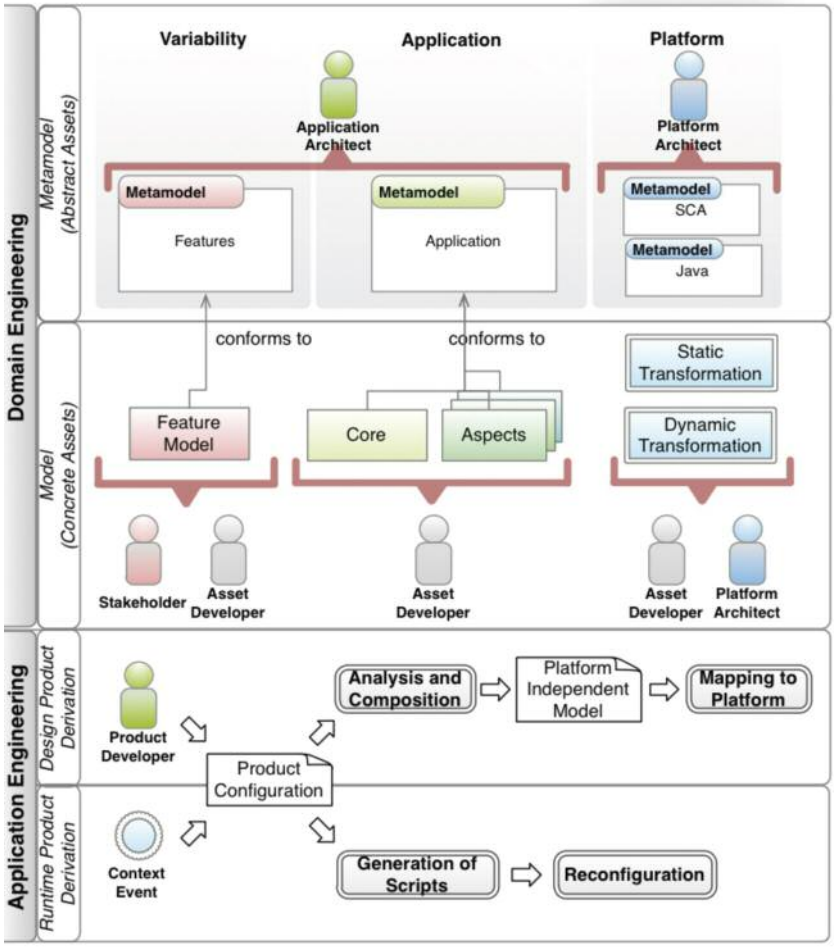


Figure 1: CAPucine software Product Line process

As a second contribution, we propose two independent processes for product derivation. Variability and aspect modelling define a complete development process that unifies the expression and manipulation of domain independent concerns at both design time and runtime. Aspect models are used in two different processes called design weaving and runtime weaving respectively. Design weaving aims at building a single product. Runtime weaving aims at adapting a product being executed. Developers can then reuse the same artifacts used for building a software product to adapt it dynamically among various configurations.

For the design weaving, the CAPucine approach is based on a model driven process where transformations and code generation are employed to obtain source code from a set of models. For the runtime weaving, we use FraSCAti, a service-oriented and component-based SCA platform with dynamic reconfiguration properties. A context manager to process events coming from the environment is associated for making decisions about the adaptation.

Link:
ADAM team
http://adam.lille.inria.fr

Please contact:
Laurence Duchien
Inria, LIFL, Université Lille1, France
Adam project-team
Tel: +33 3 59 57 78 65
E-mail: Laurence.Duchien@inria.fr

Guaranteeing Correct Evolution of Software Product Lines

by Maurice ter Beek, Henry Muccini and Patrizio Pelliccione

Researchers from the Software Engineering and Architecture group, University of L'Aquila, together with the Formal Methods and Tools group of ISTI-CNR are developing a novel approach that extends and adapts assume-guarantee reasoning to evolving SPLs in order to guarantee resilience against changes in the product environment. The proposal is to selectively verify and test assume-guarantee properties over only the components affected by the changes.

Software Product Lines (SPLs) are part of an SPL Engineering approach aimed at cost effective development of software-intensive products that share an overall reference model. Variety is achieved by identifying variation points as places in the Product Line Architecture (PLA) where specific products are built by choosing between several optional or alternative features.

While many approaches have been proposed to economize on the validation of SPL products by exploiting product similarities and proper variability management, ensuring the resilience of products of an evolving SPL is an ongoing problem. This is illustrated in Figure 1 which shows a model problem, proposed by Paul Clements and colleagues from Carnegie Mellon University.

"I run the same software in different kinds of helicopters. When the software in a helicopter powers up, it checks a hardware register to see what kind of helicopter it is and starts behaving appropriately for that kind of helicopter. When I make a change to the software, I would like to flight test it only on one helicopter, and prove or (more likely, assert with high confidence) that it will run correctly on the other helicopters. I know I can't achieve this for all changes, but I would like to do it where possible."

In an SPL context, this problem can be rephrased as: assuming various products (eg helicopters) derived from an SPL have been formally certified, what can be guaranteed for new products obtained from the SPL once one or more core components have been modified?

We took a first step towards a solution by adapting assume-guarantee reasoning, which sees the environment of a component as a set of properties, called assumptions, that should be satisfied for its functioning. If these assumptions are satisfied by the environment, then components in this environment will typically satisfy other properties, called guarantees. By appropriately combining these properties, it is possible to prove the correctness of a system before actually constructing it. Our idea is to permit selective (re-)testing and model checking of assume-guarantee properties on only those SPL components and products affected by the evolution.

Figure 2 contextualizes our work. Components used in the PLA of the SPL of interest are enriched with assume and guarantee properties (top left). The PLA configuration provides context to the assumptions and guarantees: given a component C with its assume-guarantee pair, its environment becomes the sub-architecture connected with C. The process is complicated by the fact that the reasoning is performed on the PLA, including all variation points, rather than on each indivdual product. This means that the assumptions need to be calculated in a smart way taking into consideration the variability management of the components. Once a component evolves (top right), this modifi-

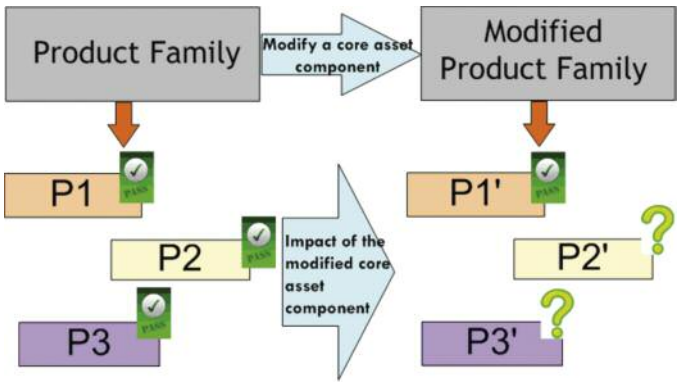


Figure 1: Evolution requires quality re-evaluation.

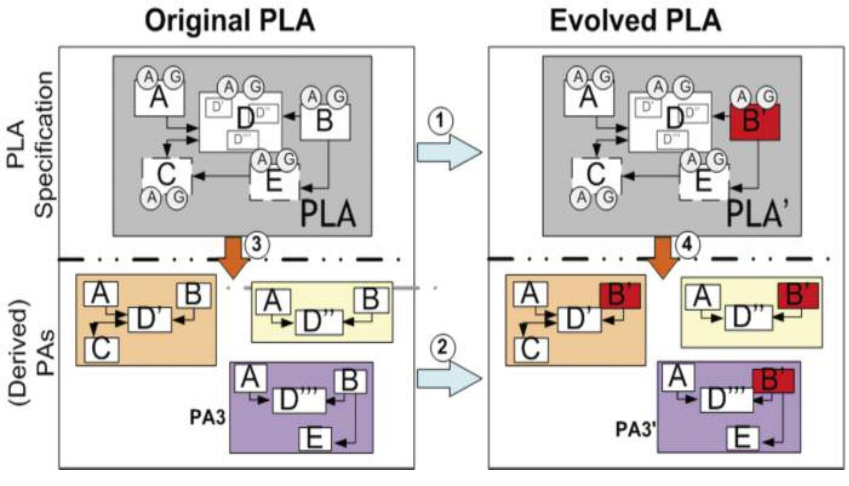


Figure 2: Contextualization of our work

cation is expected to have an impact on several Product Architectures (PAs), namely each PA that contains the modified component. For instance, when B evolves into B', the assumption and guarantee pairs of both B and B' must be checked.

Our solution thus envisages a combination of regression testing and assume-guarantee testing applied to evolving PLAs. Conceptually, this requires us to better understand two issues.

First, how to extend assume-guarantee testing to evolving architectures. The assume-guarantee pair associated with each component in an architecture is normally used to generate component-specific testing traces which, once evaluated against the appropriate assume-guarantee premise, can show whether the composition of components can produce failures. Assuming a PLA/PA has been tested, the challenge becomes how to apply assume-guarantee reasoning for regression testing the modified PLA/PA (see (1) and (2) in Figure 2, respectively). Second, how to use the relationships between a PLA and its PAs (see (3) in Figure 2) to apply regression testing to the evolved PAs.

In summary, we currently envisage what we call a double regression testing approach, in which an evolved product (eg PA3' in Figure 2) can be tested based on how it regressed from PA3, and based on its relationship with the architecture of its family, PLA' (cf. Figure 2). Our expectation is that this considerably reduces the effort needed to re-analyze products of an SPL upon its evolution.

We are working out the details of our approach, after which we intend to implement it and investigate its effectiveness by applying it to case studies.

Please contact:
Maurice ter Beek
ISTI-CNR, Italy
E-mail: maurice.terbeek@isti.cnr.it

Software Evolution in Model-Driven Product Line Engineering

by Silvia Abrahão, Javier González-Huerta, Emilio Insfran and Isidro Ramos

New requirements and technology changes lead to continuous changes of the assets comprising a software product line. Since the product line represents a large number of potential products (or already deployed products) in a given domain, managing these changes becomes a key issue when dealing with evolution. We present a framework to support the development and evolution of high-quality software product lines. The framework is based on several interrelated models or system views (eg, functionality, variability, quality) and a production plan defined by model transformations that generate a software system that meets both functional and quality requirements. We used our framework to develop a software system for the automotive domain.

Software Product Lines (SPLs) are families of products that share common functionality but also have variations tailored for different customer needs. Many of the benefits expected from SPLs are based on the assumption that the additional investment in setting up a product line pays off later when products are created. However, product line assets have to evolve continuously in order to keep the economic benefits of a product line at an optimal level. Managing this evolution therefore becomes a key issue when maintaining a product line.

In the MULTIPLE project, we defined a framework to support the development and evolution of high-quality SPLs. This framework is based on the definition of a multimodel that represents the different system views and the relationships among them. The variability management involves the manipulation of features, represented as cardinality-based feature models, and the support of such variability in the so-called product line core assets. Specifications of the system variability, functionality, and quality can be dealt with models that are independent from each other but where their inter-consistency is assured by means of the relationships defined in this multimodel.

The SPL production plan is parameterized by means of the multimodel which specifies the corresponding model transformations that are needed to obtain a specific product with the desired features, functions, and quality.

Figure 1 shows the multimodel playing a pivotal role in the SPL production plan. In the domain engineering phase it is used to express the impact and constraints among features, functional components and quality attributes, describing in this way, the extension of the product line. In the application engineering phase, it guides the product configuration, allowing the selection of features and functional components that meets the quality attributes selected by the application engineer. This approach leads us to re-consider the problems related to the intra (eg internal consistency of the feature model) and inter-model consistency (eg correspondences among elements of the feature and quality models) in a broader and realistic context.

Model-based evolution of software systems implies the evolution by using models, eg applying model-driven techniques to support product evolution, or the evolution of models, ie the evolution of the models/metamodels that describe the product. In our framework these evolutions are done by means of reification of software artifacts at the metalevel. When the evolution affects the types then the metamodels need to be reified and evolved. A reflection process translates to the source level the evolved artifact. At a model level, the process is similar. The models are reified at the metalevel, evolved, and then

reflected. In the MOF hierarchy, the evolution behaves in a similar way in each level and the consequences over the instances of the evolved artifact can be computed automatically.

Figure 2 shows the main steps of a production plan to develop a software system by applying model transformations for the automotive domain. The quality attributes that the product must fulfill are used as the decision factor when alternative transformations appear. In this way, the application engineer can experiment with different quality-driven model transformations (which represent design alternatives) and choose the one that better satisfies the functional and quality requirements.

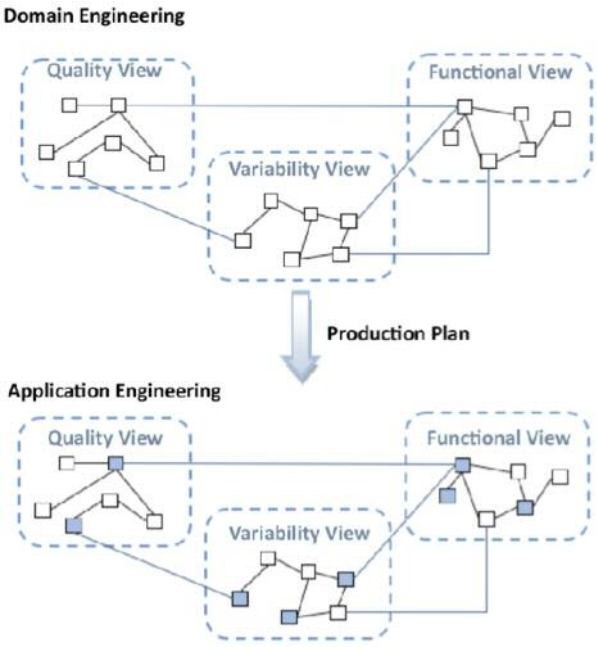


Figure 1: The Multimodel in the Software Product Line development process

The output of the process is the product represented by means of its architectural models. These architectural models can be completed in successive model transformation processes to generate the fully functional version of the product. In addition, in each step of the process, the quality model can be used to determine whether the artifacts obtained (eg product architecture) meet the expected quality levels by automatically applying metrics defined on the quality model. This assessment provides evidence about the correctness of the individual models as well as the multimodel relationships.

Finally, the multimodel helps to establish the traceability and to ensure the consistency among the system views. Furthermore, the model transformations are guided by the structural constraints established by the multimodel and the quality view contributes to the quality assurance and improvement of the product line artifacts. Currently, we are building a tool to give support to this approach.

Links:
ISSI Research Group (Universitat Politècnica de València)
http://issi.dsic.upv.es/projects

FMCL: Feature Modeling Constraint Language
http://issi.dsic.upv.es/~agomez/feature-modeling

Please contact:
Silvia Abrahão
Universitat Politècnica de València, Spain
Tel: + 34 963877000
E-mail: sabrahao@dsic.upv.es

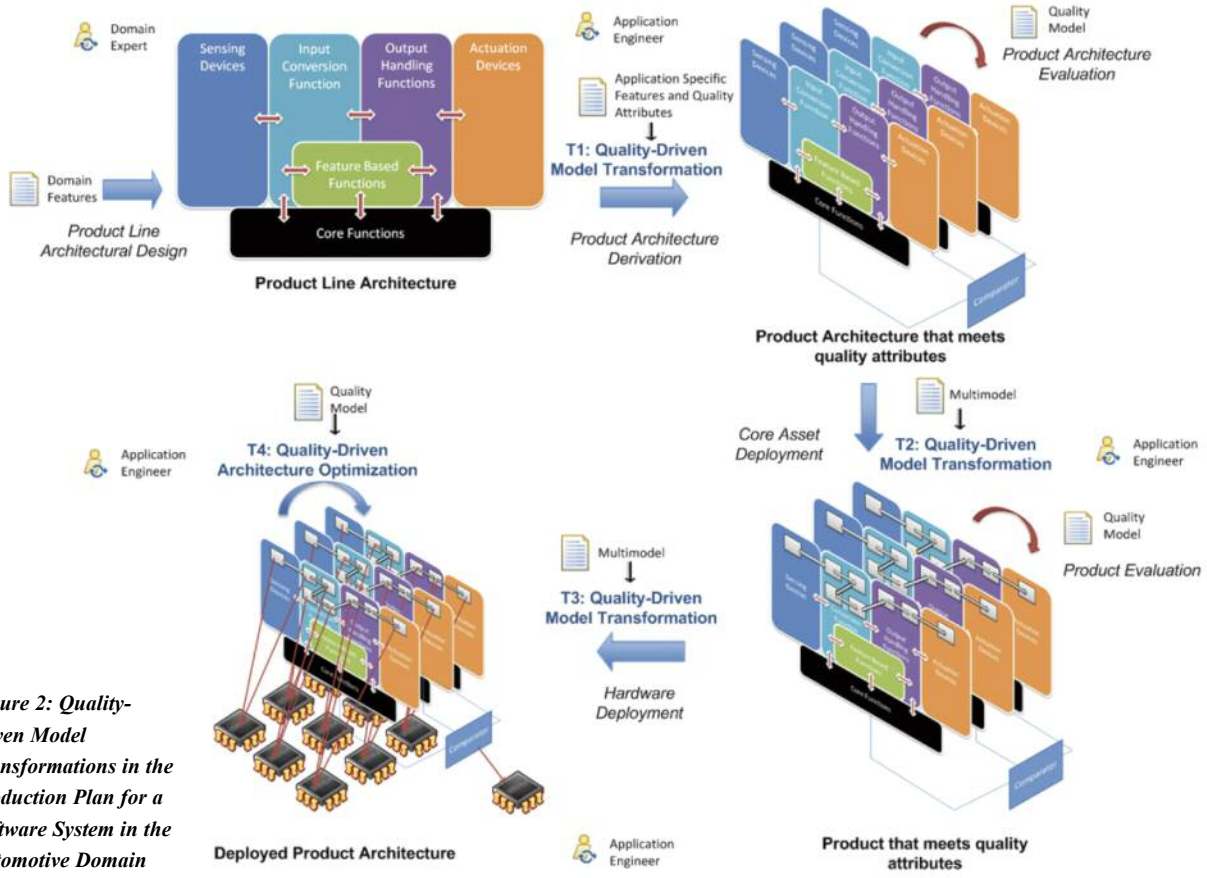


Figure 2: Quality-driven Model Transformations in the Production Plan for a Software System in the Automotive Domain

How to Deal with your IT Legacy: What is Coming up in MoDisco?

by Hugo Bruneliere, Jordi Cabot and Grégoire Dupé

The Eclipse-MDT MoDisco open source project is part of the Indigo Eclipse Simultaneous Release. Here we describe how MoDisco can play a role in the evolution of (legacy) software, focusing on the latest project news.

MoDisco is an official Eclipse-MDT project, devoted to the Model-Driven Reverse Engineering (MDRE) of IT systems, which has already been successfully applied to the software evolution and modernization of many industrial legacy systems.

We believe that dealing with legacy systems is one of the main challenges of the software industry today. Indeed, we see increasing numbers of "revolutionary" technologies popping up which create tensions with all the "old-fashioned" existing systems which are still relevant and used within companies: to be able to adopt the new, you need to get rid of the old (or at least hide it).

Having a better understanding of legacy systems in order to document, maintain, improve or migrate them is not an easy task but is a key requirement in many companies. MoDisco intends to facilitate the process by offering a reusable and extensible model-based framework that facilitates the creation of reverse engineering solutions for software understanding, evolution and modernization. Relying on different Model Driven Engineering (MDE) technologies provided in Eclipse, all based on the Eclipse Modeling Framework (EMF), it provides a set of concrete components that can be combined efficiently to tackle this important issue.

The MoDisco open source initiative started in 2006 when the AtlanMod Team (Inria, Ecole des Mines de Nantes & LINA) and Mia-Software (Sodifrance Group) were working together in the context of the IST-FP6 MODELPLEX European project. AtlanMod created the MoDisco project in Eclipse-GMT (Generative Modeling Technologies, which acted at the time as an incubator for modelling prototypes), and several experimental components were then developed and contributed. Mia-Software quickly started contributing to the project and officially joined MoDisco in 2008. Since then, the MoDisco community grew pro-

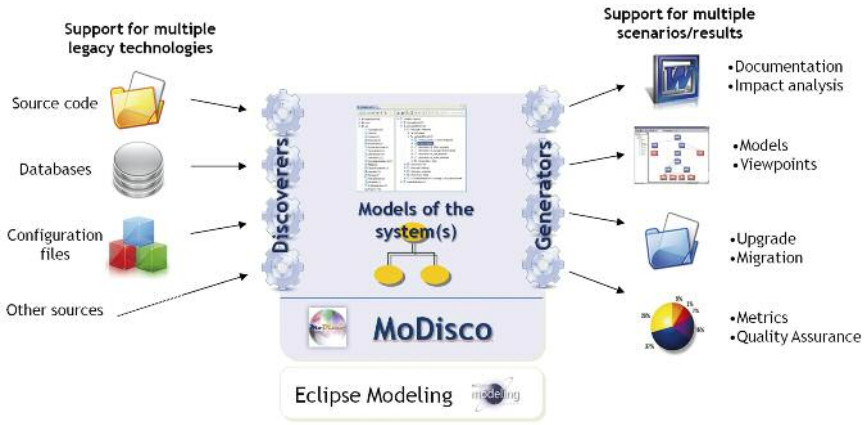


Figure 1: MoDisco, a Model Driven Reverse Engineering approach & framework

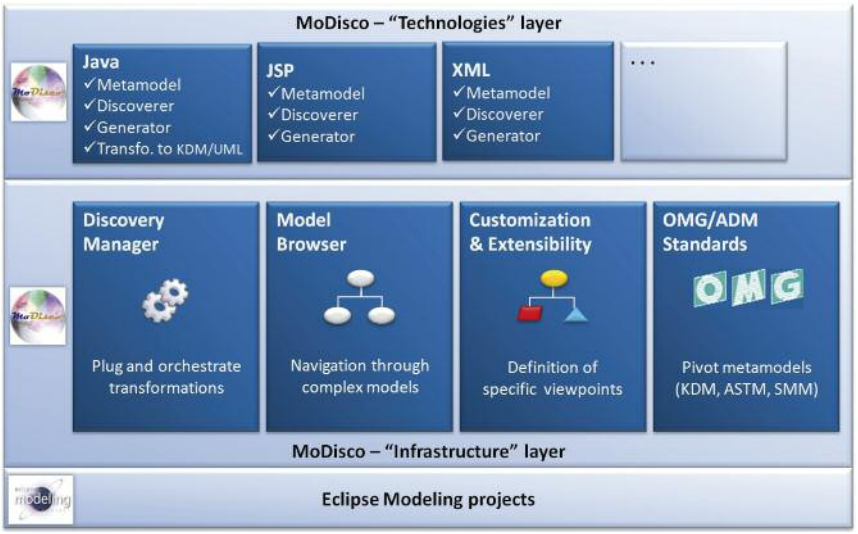


Figure 2: The MoDisco overall architecture & provided components

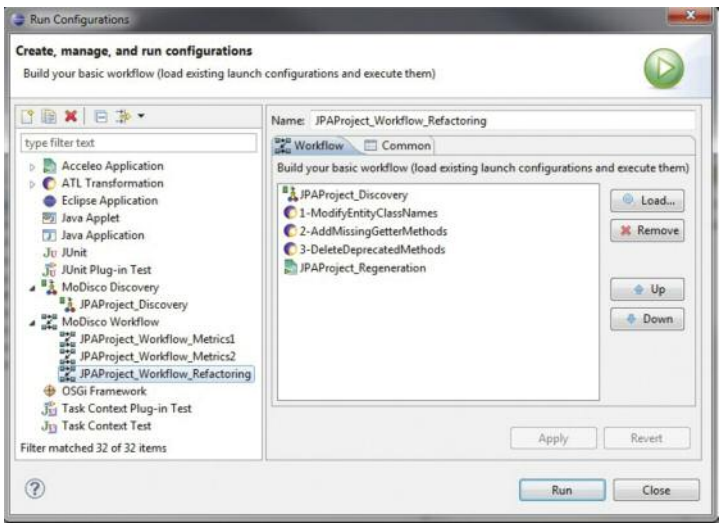


Figure 3: A sample Java refactoring workflow implemented with MoDisco

gressively to reach its current high maturity level. The MoDisco base infrastructure and set of components are settled and, consequently, the project has been naturally promoted to Eclipse-MDT (Model Development Tools) which now has solid industrial support.

As shown in Figure 1, MoDisco can be effectively applied for implementing evolution and modernization processes (technology migration, software refactoring, architecture restructuring, etc), as well as for retro-documentation, quality assurance and any other situation in which a better understanding of existing IT systems is required.

Owing to its modular architecture (see Figure 2), the MoDisco framework encompasses the three steps of a standard MDRE approach: 1) Discovery (ie extracting a complete model of the source code), 2) Understanding (ie browsing and providing views on this model for a given purpose) and 3) Transformation (evolving the model towards a new technology/architecture etc).

More specifically, as part of its "Infrastructure" layer, MoDisco offers a set of generic (ie legacy technology-independent) reusable components that play an invaluable role in building the core of MDRE solutions: Discovery Manager & Workflow for MDRE task orchestration, Model Browser for advanced navigation in complex models, model extension and customization capabilities for understanding (eg views definition), etc.

As part of its "Technologies" layer, it now provides an advanced support for the Java, JEE and XML technologies, including complete metamodels, corresponding model discoverers, transformations, code generators, customizations, query libraries, etc. Figure 3, for instance, shows a sample Java refactoring workflow in the MoDisco tooling.

MoDisco is also considered by the OMG Architecture Driven Modernization (ADM) Task Force as the reference provider for real implementations of several of its standards: Knowledge Discovery Metamodel (KDM), Structured Metrics Metamodel (SMM) and Abstract Syntax Tree Metamodel (ASTM). A complete documentation detailing all these components, as well as the project and related support in general, is freely available from the MoDisco project web site.

Today, the MoDisco approach and underlying framework are used as a solid base for real applications within companies. However, some important improvements on MDRE techniques are still needed. Important research issues to be addressed in the near future within the context of MoDisco include:

- Scalability of model manipulation techniques (loading and unloading, discovery, querying, transformation, etc);
- Advanced composition of heterogeneous models;
- Traceability during the whole life cycle of a MDE/MDRE project.

Advances in these areas will strengthen the capabilities of the MoDisco framework, helping us to achieve our main objective: to provide the modernization engineer with a powerful and reliable approach, platform and corresponding support for elaborating on complex large-scale reverse engineering solutions.

Links:
MoDisco:
http://www.eclipse.org/MoDisco/
AtlanMod Team:
http://www.emn.fr/z-info/atlanmod
Mia-Software:
http://www.mia-software.com/
MODELPLEX project:
http://www.modelplex.org/
ADM Task Force: http://adm.omg.org/

Please contact:
Hugo Bruneliere, Jordi Cabot
Inria, Ecole des Mines de Nantes, France
AtlanMod team
E-mail: hugo.bruneliere@inria.fr, jordi.cabot@inria.fr

Grégoire Dupé
Mia-Software (Sodifrance Group), France
E-mail: gdupe@mia-software.com

Model-driven Evolution for Multimodal Mobile Geographic Information Systems

by Nadia Elouali, Daniel Liabeuf, Xavier Le Pallec, José Rouillard and Jean-Claude Tarby

In the MOANO project we aim to create an end-user modelling environment to design and generate multimodal mobile Geographic Information Systems (GIS) capable of evolving over time. Here we discuss different challenges and suggestions to address them using a Model-Driven Engineering (MDE) approach.

The use of mobile devices is very widespread. Mobile devices are communication tools that give users access to different computing services and are generally equipped with various sensors. Many applications take advantage of these attributes and consequently some new uses have emerged. The portability and precise positional information offered by modern mobile GIS means that they are far more user-friendly than traditional devices. However, there is still much scope for the development of 'good practices' in the use of mobile GIS, for example the best fit of needs or more natural interactions still requires much work.

In order to accelerate the development of these 'good practices', we propose through the MOANO project (Modèles et Outils pour Applications NOmades de découverte de territoire - 'Models and Tools for Pervasive Applications focusing on Territory Discovery') to allow end-users to build their own mobile GIS and evolve them over time using a dedicated modelling environment. Botany was chosen as our investigated field since botanists need mobile

Figure 1: Using the augmented reality in the botanical garden

GIS in their daily tasks to access their noted observations and botanical information (Figure 1). The project also addresses multimodal interactions to overcome the limitations of mobile devices and to fully exploit their sensors. Multimodal interactions are user interactions through different modalities such as gesture and voice and the capability to use them simultaneously or alternately.

'Good practices' will be developed with the help of input from the botanists involved. First they will create their multimodal mobile GIS according to their initial needs. Then they will identify the different use constraints and evolve their systems according to the identified new needs. Thus, the 'good practices' of multimodal mobile GIS shall be defined and used over time by users themselves with less computer engineer involvement.

We use a MDE-based approach for this project since it utilizes models in the first stage of development. The main benefit of models is their use of visual representations which are a good communication tool (to be more relevant, modelling should be performed by different users). In addition, they are not technology-dependent (for languages or frameworks for instance), thus botanists can create their own business process models at a high abstraction level.

In order to create our desired MDE-based modelling environment we explored the following challenges:

- *Multimodal interaction modelling*
  Multimodal interaction design and development have always been considered to be difficult tasks. This is due to the need to support various interaction modes, the different input and output devices and the required set of recognition technologies. In addition, the combination of modalities introduces other challenges related to time and event management. We propose the use of a multimodal modelling language to tackle these challenges. Current languages are generally limited to one direction (input or output) whereas we consider both directions in our project. In addition, we need a user appropriate modelling language targeting botanists. Thus a new end-user modelling language is required to design the multimodal interaction.

- *The domain modelling*
  We also need to define a language for botanists to express the software solution that they intend to use. This language must offer enough concepts to cover their needs, offer sufficient benefits to be worth using and be accessible for end-users.

- *Concept presentation*
  The visual concept notation must be chosen according to the botanist's profile (for instance gardener or manager). Its cognitive efficiency can be evaluated using many endpoints such as semiotic clarity, ie using one symbol to represent a given concept (vocal modality must be represented by two icons since it is used in input and output). But how can we evaluate the clarity? Indeed, the quantitative evaluation of the model's quality is currently a problem by itself.

- *Interpretation and evolution of the model*
  The modelling environment should provide the correct interpretation of the botanist's models and allow their evolution. The major evolving challenge for the model is to maintain traceability links between model elements and to ensure consistency with user's specified evolutions. In addition, another challenge is related to the model's interpretation using multiple visual notations since they change according to the users profile.

We started our project by collaborating with botanists through our partner in MOSAIC Park (Figure 2). Currently, we are addressing the challenges outlined here by offering the botanists a modelling tool based on the Wizard of Oz experiment.

Figure 2: The MOSAIC Park

Link:
http://moano.liuppa.univ-pau.fr/

Please contact:
José Rouillard
Laboratoire d'Informatique
Fondamentale de Lille (LIFL)
University of Lille, France
Tel: +33 3 20 33 59 37
E-mail: jose.rouillard@univ-lille1.fr

EVOLIS: A Framework for Evaluating Evolution of Information Systems

by Alexandre Métrailler and Thibault Estier

This research proposes a framework (named EVOLIS) to study information systems (IS) evolution. The benefits of the EVOLIS framework are twofold. First, it gives managers a tool for assessing the impact of a change from either the user or the business/IS perspective. This can be used to design a strategy for IS evolution. Second, the repeated use of the EVOLIS framework reveals specific evolution patterns.

The rapid change occurring in business environments in response to evolving markets leads to a considerable amount of change in business processes. In order to cope with changes and new market opportunities, the information systems (IS) that support business processes need to be able to evolve.

The term evolution, in relation to software systems, has various interpretations depending on the role of stakeholder. When we refer to IS evolution we mean a process of discrete progressive change over time in architecture, workflows, features or functionalities of IS. For instance, an ERP system typically evolves by regularly adding new transactions, processes and views on business processes during its life cycle. The different stages of maturity successively provided by an e-banking application to customers is another example of IS evolution over time.

The goal of this research is to create a framework to evaluate the effects of IS evolution. In other words it must be able to characterize the impact of changes according to predefined criteria in order to help managers to refine their IS strategy. The framework design should cover the following specifications:

- it should be easy for managers to understand and use
- it should address users' perceptions of the system and their efficiency using the system
- it should take into consideration objective factors such as the maturity level of technology used, the alignment of the solution with the business and its level of integration.

The EVOLIS framework exists in the form of a canvas consisting of 4 blocks: IS/Users Fit, Technology, IS Integration and Alignment with the business.

The IS Fit with Users is measured using both subjective and objective approaches. The users' satisfaction with the IS is measured using the well known perceived usefulness and perceived ease of use (TAM). These two variables determine whether users accept or reject an information technology. Perceived usefulness is defined as the degree to which a user believes that using the modified system would enhance job performance. Perceived ease of use is described as the degree of ease with which a person uses the modified system. Consequently, these two variables enable us to measure the perceived benefits or losses of a change. On the other side, the objective performance of users is calculated using measures such as efficiency and effectiveness to perform tasks.

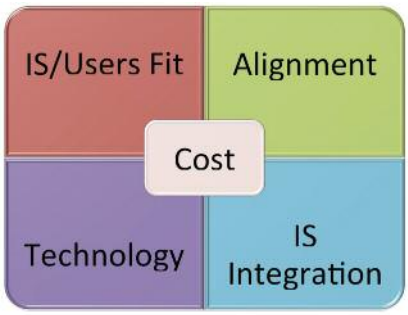


Figure 1: The 4 building blocks of the EVOLIS framework in parallel with the cost of IS

The Technology, IS integration and Business/IS Alignment blocks provide information on how well the IS supports the business. The Technology block represents the degree of innovation, anticipation, flexibility, and scalability of the IS. The IS integration block evaluates the level of integration of the IS. It measures the delta between the past IS and the changed IS. There are different types of IS integration evolution, namely an evolution of integration among components of the system, among business functionalities, or an integration with systems outside of the company, etc. The Business/IS Alignment block describes the fit between business processes and IS processes. It also reflects on the scope of the IS, whether it is extended by the evolution and whether the evolution addresses core business functionalities or support functionalities.

These 4 blocks must be evaluated in parallel with the cost function of the IS. This provides an evaluation as to whether the ROI (return on investment) is satisfactory or not.

The use of the EVOLIS framework after each evolutio n of the system pro-

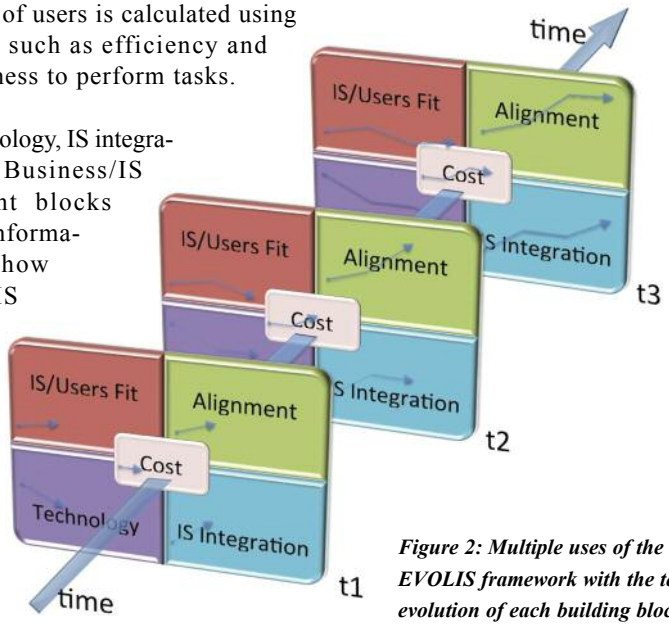


Figure 2: Multiple uses of the EVOLIS framework with the temporal evolution of each building block

vides a temporal view of system evolution. It acts as an indicator to determine in which direction the system should evolve. The use of EVOLIS also helps managers to design the strategy of IS evolution. Moreover, a temporal view clearly identifies specific evolution patterns of the IS.

This research is conducted using a design science approach in IS. The design and the development of the EVOLIS framework is an iterative process based on case studies and practitioner interviews to refine and demonstrate the use of this framework. The evaluation of EVOLIS will be qualitative, principally based on practitioners' feedback and satisfaction surveys. We will determine whether IS managers are willing to adopt the EVOLIS framework to evaluate the evolution of their IS and to use it as part of their IS strategy.

This research originated from a PhD project in the Information System Institute of the Faculty of Business and Economics at the University of Lausanne. The work, which commenced in 2010, is ongoing.

Please contact:
Alexandre Métrailler
University of Lausanne, Switzerland
Tel: +41 22 692 35 86
E-mail: alexander.metrailler@unil.ch

Mathematics Meets Chemistry - Workflow-guided Evolving Software for Molecular Modelling

by Dirk Reith and Karl Kirschner

Computational chemistry began with the birth of computers in the mid 1900s, and its growth has been directly coupled to the technological advances made in computer science and high-performance computing. A popular goal within the field, be it Newtonian or quantum based methods, is the accurate modelling of physical forces and energetics through mathematics and algorithm design. Through reliable modelling of the underlying forces, molecular simulations frequently provide atomistic insights into macroscopic experimental observations.

There is great demand for well coded software that incorporates new technology on a regular basis. Since increasing numbers of atoms are needed to simulate increasingly complex systems, computing resources can easily be a limiting factor for scientists in this field, hence there is a real need for efficient software. Although there are corporate-based developments, most software packages are written by natural scientists rather than software engineers. This is due to the demanding physical and chemical concepts that need to be transferred into proper algorithmic solutions. Both academic and open-sourced development, plus some corporate software, tend to be POSIX command-line based, whose input and output can be easily parsed by researchers. We exploit this command line and parsing ability in designing new software tools that are tailored to specific research needs. Our goal is to take advantage of the strengths of various existing third-party software by creating links between them, and subsequently develop new algorithms that allow us to incorporate state-of-the-art ideas. This demands that our software should evolve to a) communicate with new releases of third-party software and b) to incorporate new ideas presented in the primary research literature.

Our solution for enabling our software to evolve is to decouple tasks within the software. In doing so, algorithmic solutions can be introduced in a modular fashion, allowing us to easily identify and update specific tasks as needed. Using these ideas, our group has developed two independent software tools, called WOLF$_2$PACK and GROW. A third tool called ESPResSo++ has been developed in collaboration with the Max Planck Institute for Polymer Research (MPI-P) in Mainz. Taken together, these tools enable us to quickly investigate a diversity of chemical and biochemical problems. Figure 1 shows that they serve on different, but inter-linked, resolutions of molecular modelling. A brief summary of each program follows.

The quality of results from molecular simulations is heavily dependent upon the quality of the underlying molecular parameters. While very important, this

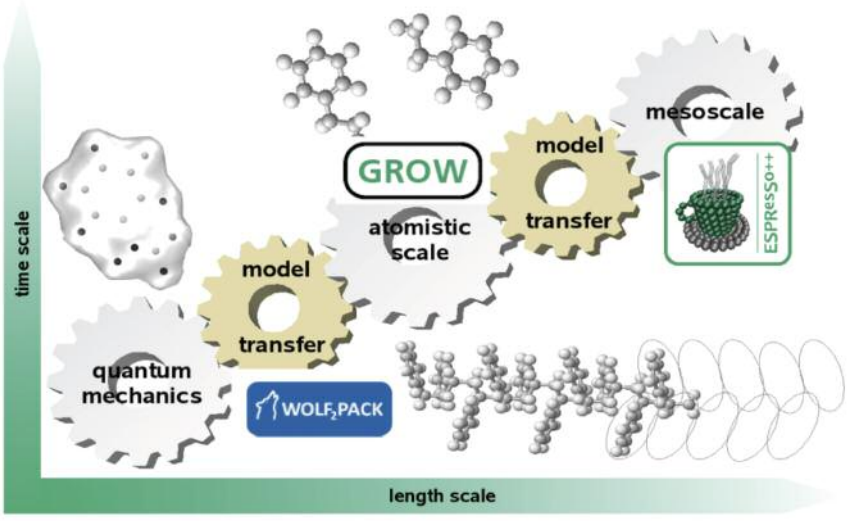


Figure 1: Current evolving software packages (co-)developed at Fraunhofer SCAI and how they fit into the multiscale modelling approach of computational chemistry.

requires specialized knowledge and is very time consuming. Worse, the procedure differs for intramolecular coordinates (which can be derived from quantum mechanics(QM)) in comparison to the intermolecular interactions (which can only be matched versus macroscopic observables as measured by experiments). Therefore, one of the primary goals of our research is to develop accurate and reliable molecular parameters in a reasonable time and as error free as possible.

For intramolecular interactions, we have created a scientific "Workflow for force-field optimization package" ($WOLF_2PACK$) that incorporates our beliefs for how QM-gained knowledge should be transferred to Newtonian-based models. We define a scientific workflow as a series of independent steps that are linked together according to the data flow and the dependencies between them. For intermolecular interactions we developed a systematic optimization workflow, based on efficient gradient-based numerical algorithms called GROW. GROW is a modular tool kit of programs and scripts. It is a generic implementation and can be easily extended by developers. Both tools are written by natural scientists and software engineers to really make mathematics meet chemistry. They facilitate the a) development and optimization of molecular parameters for a given simulation engine, b) transfer of parameters from one software package to another, and c) testing of the parameters using a standard test suite and protocol via an semi-automated iterative parameterization process.

Thirdly, the Extensible Simulation Package for Research on Soft matter systems (ESPResSo++), jointly developed with the MPI-P, is a parallelized, object-oriented simulation package designed to perform many-particle simulations of molecular chemical systems. The workflow idea is realized in ESPResSo++, firstly, in its model builder that allows scientists to easily create chemical systems in a flexible manner. Secondly, the modular design is created along the physics of the systems, allowing for the efficient addition of new physical effects through algorithms. The realization of both aspects is also aided by the separation of the algorithmic kernel and a simulation design and control front end. The ESPResSo++ kernel ensures efficiency through the use of advanced C++ programming language features, high-performance storage techniques, and cache optimization while users steer and control simulations by a Python script. This approach makes it easy to conduct online analysis and interactive simulations. The program structure should enable scientists to use ESPResSo++ as a research platform for their own methodological developments, which at the same time allows the software to grow and acquire the most modern methods.

Taken together, the joint efforts of natural scientists and software engineers greatly enhance software development for molecular modelling. The concept of evolving software seeded into the code development itself is a key for maintaining state-of-the-art tools and research.

Links:
http://www.scai.fraunhofer.de/coche.html
http://www.espresso-pp.de

Please contact:
Dirk Reith
Fraunhofer Institute for Algorithms and Scientific Computing (SCAI), Germany
Tel: +49 (0) 2241 14 2746
E-mail: dirk.reith@scai.fraunhofer.de

Fostering Collaboration in the Modelling, Verification and Evolution Research Communities in Belgium

by Carlos Noguera, Andy Kellens and Theo D'Hondt

For more than 20 years, the Belgian Science Policy has been promoting the Interuniversity Attraction Poles (IAP) programme, with the aim of providing an incentive for the formation of excellent research networks. As a means to achieve this goal, the IAP programme focuses on fostering collaboration between research groups, rather than on research results. It is within this programme that five years ago the "Fundamental Issues in Software Engineering: Modelling, Verification and Evolution of Software" (MoVES) project was founded. We report here on how this project was able to both advance the state-of-the-art in the domain of software evolution, and form a cohesive national research network.

The network is composed of teams from eight Belgian universities, and collaborates explicitly with two international teams - the ADAM team from Inria (France) and the SERG group at TU Delft (Netherlands). Before the project started, Belgian research in modelling, programming languages and verification was done in relative isolation, whereby research teams collaborated more closely with international partners than with "intra-national" ones. Thus, the IAP programme provided an ideal motivation for closing the academic ties of teams working around the related domains of modelling, verification and evolution of software. To foster such collaborations, a thematic approach centered around three main axes, and structured around seven work packages was established.

The three axes around which the research activities of the MoVES network are structured are: Programming and Modelling languages, Model Analysis, and Model Evolution. The

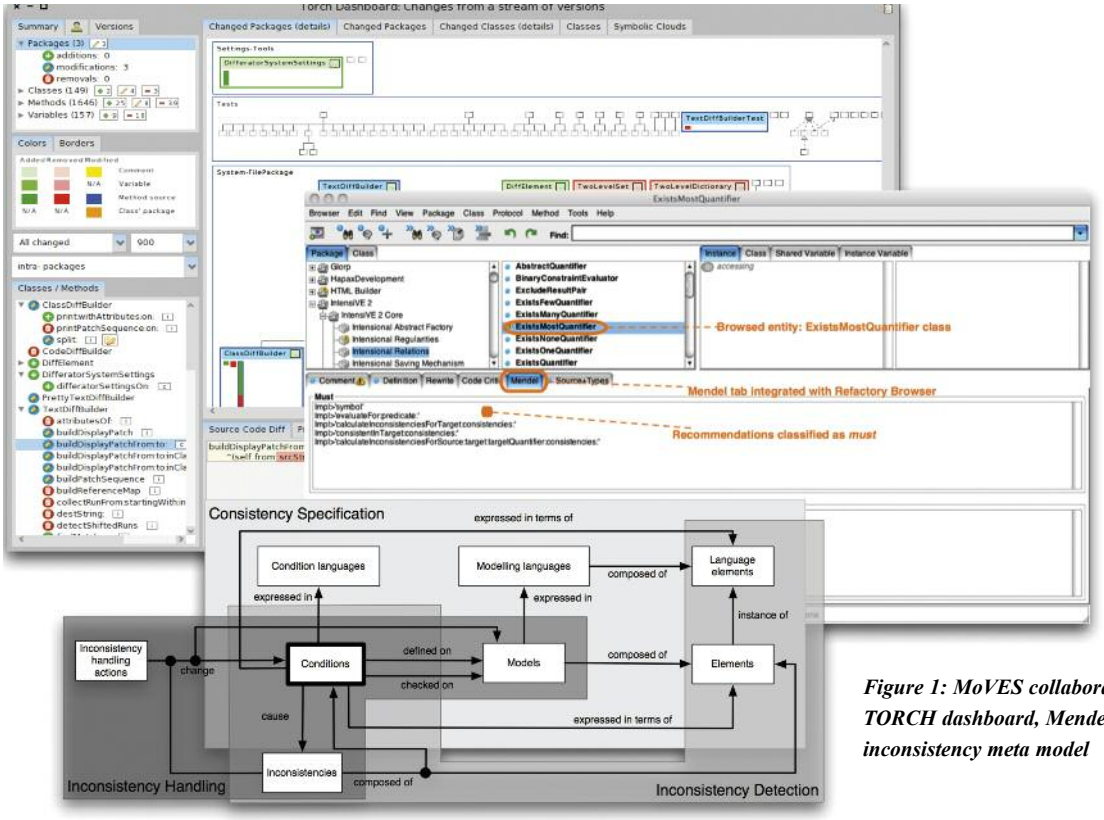


Figure 1: MoVES collaborations: the TORCH dashboard, Mendel and the inconsistency meta model

Programming and Modelling Languages axis defines, extends and formalizes programming and modelling languages that can be processed automatically to analyse certain software properties or to reason about the evolution of programs and models expressed in those languages. The Model Analysis axis studies a variety of techniques and tools to compose and transform models, to check their consistency and to verify their properties. Finally, in the Model Evolution axis we study techniques to incrementally repeat those analyses and transformations as models and programs evolve, to keep models that evolve independently consistent with one another, as well as techniques to restructure existing models while preserving their intended behaviour and other desired properties.

Three success stories attest to the value of MoVES and would not have been possible without the support of this collaboration-oriented research network. All three examples are taken from the activities of the network related to software evolution.

First, the network promotes the joint supervision of PhDs as a means for establishing lasting collaboration. So far, three students are participating in co-tutelles, two of them between Belgian universities; the third between a Belgian university and Inria. The topic of the latter being "Supporting Source Code Changes". The main goals of the thesis are to provide means to represent and query the history of open-source code, to characterize and aid in understanding such history, and to provide explicit support for dealing with its integration. As a result of this co-tutelle, we have seen the link between the two participating teams strengthened, resulting in various other collaborations and researcher exchanges.

Second, work package topics have been leveraged to jointly tackle research problems. Singular amongst them lies the work of work package 4 on consistency checking and co-evolution. In this work package, the development of a unifying conceptual framework for inconsistency management approaches is attacked as a work package-wide effort. This is a non-trivial and as yet unfinished research achievement that actively involves all partners in the consortium, with regular joint meetings to discuss the progress. The experiment of trying to coordinate previously individual efforts has brought the participants closer together. For example, this has resulted in obtaining a five-year national research project involving the participants of the work package.

Third, we have observed numerous cases throughout the network where fleeting collaborations between researchers evolved into a timely and lasting cooperation. One example of such a collaboration -- between a Flemish and a Walloon partner of the network -- can be found in the topic of "Mining and Validating Structural Design Regularities". While the involved researchers were introduced via the MoVES network and decided to team up to investigate a single idea, namely the use of association rule mining to extract latent information from source code, this collaboration has outlasted the investigation of this initial idea and resulted in a long-term and structural collaboration that culminated in various joint publications.

As the current phase of the IAP Programme draws to a close, we have submitted a proposal for extension of the MoVES project. Regardless of whether the project is selected for continuation, we believe that the links forged during the past five years will persist and continue to flourish.

Please contact:

Carlos Noguera, Andy Kellens, Theo D'Hondt
Software Languages Lab,
Vrije Universiteit Brussel, Belgium
E-mail: cnoguera@soft.vub.ac.be,
akellens@vub.ac.be,
tjdhondt@vub.ac.be

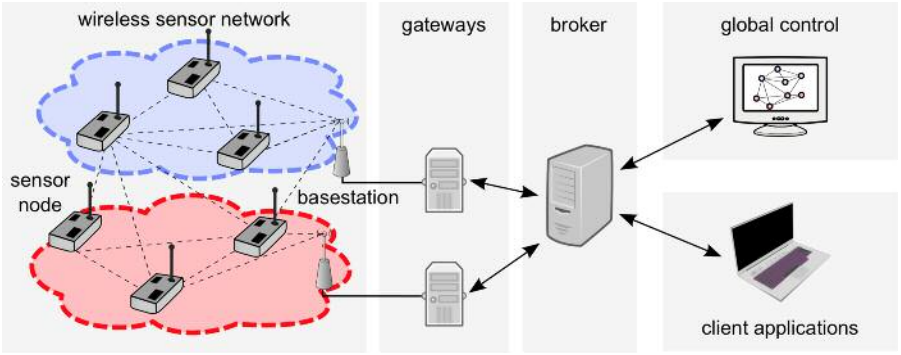


Figure 1: A centrally controlled wireless sensor network architecture. The architecture includes the wireless sensor network divided into clusters, each managed by a gateway, a broker to handle the communication between the gateways and the global control as well as to sensor applications.

A Wireless Sensor Network that is Manageable and Really Scales

by Urs Hunkeler, Clemens Lombriser and Hong Linh Truong

This wireless sensor network (WSN) was motivated by the need for a manageable and scalable network constructed from relatively inexpensive off-the-shelf hardware. The WSN consumes very little power. The resultant network is in operation in several applications with up to 100 nodes. Simulations have shown excellent performance with 500 nodes. The network features centralized (rather than distributed) control; this aspect is the theme in this short paper.

Centralized rather than distributed control

It is an almost universal belief within the WSN community that centralized management of large networks has to be inefficient. We have encountered this statement in many publications - but without proof.

One of the main arguments quoted against centralized algorithms is that a global view of the network needs to be obtained before the network can perform its work. Collecting this information is considered to scale badly and to involve a substantial amount of communication and thus energy loss. Usually neglected is the fact that distributed algorithms initially require extra effort as well - and need time until they settle into a stable configuration. We argue that with an efficient topology-discovery mechanism the impact of establishing a global network is reduced far enough to make a centrally managed solution competitive.

Another argument for distributed algorithms is that they are more flexible in adapting to changes in the network topology, such as when nodes fail or link qualities change. Local decisions by the sensor nodes can quickly adapt and correct the problem, while loosing only little data. A centralized algorithm, however, can as well take care of such situations and instruct the network to act accordingly and in a coordinated fashion. Furthermore, having a single entity taking decisions improves the controllability and observability of the adaptation.

In many cases it is also possible and even desirable to detect bad connectivity already at deployment time, such that appropriate physical changes can be made and the WSN does not have to cope with too many bad links at all. We believe that it is in general better to fix bad connectivity with physical modifications at deployment time than to try to capitalize on short-term improved link qualities.

It is also often argued that a centralized architecture introduces a single point of failure, namely the basestation controlling the whole network. In many cases, however, it is the basestation that receives all the data from the network, and if it fails, the WSN cannot export its data anyway; the WSN becomes useless. In a centrally controlled approach, sensor nodes may be able to detect this situation much faster than with a distributed protocol. The centralized control can then take appropriate measures such as going into a low-power waiting mode. If a distributed approach is badly designed, it may continue to run for a considerable amount of time while pointlessly wasting energy in trying to find working basestations.

Further advantages over distributed solutions:

- Simplicity of the code in the wireless nodes reduces cost and chances of malfunction.
- Needing to store little information in the nodes further reduces cost.
- Observability and controllability are enhanced by having all the information at a single point.
- Network management is eased in the sense that expert knowledge can be performed by the central control to optimize the operation of the network.
- Any sophisticated algorithm, even a distributed one, can be applied to determine the network configuration before deployment.

Additional features (covered in the referenced, more extensive report)

Taking advantage of the centralized control and adding to its effectiveness are a number of novel features, themselves novel contributions to the state-of-the-art. These include:

- an efficient algorithm for multi-hop network topology discovery and link quality estimation,
- a flexible TDMA superframe structure for supporting an ultra low power mode of operation,
- a combined clustering and routing algorithm for partitioning the complete WSN into multiple clusters with a-priori defined basestations, and
- a scheduling algorithm for multi-cluster and multi-channel data collection incorporating message aggregation.

Results for a 29-node field test are also included in the report.

Link:
http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5984921

Please contact:
Hong Linh Truong,
IBM Zurich Research Laboratory, Switzerland
E-mail: hlt@zurich.ibm.com

Wireless Sensor Networks and the Tower that Breathes

by Luca Mottola

The next Internet, Internet of Things, allows for unprecedented levels of interaction between the digital and physical worlds. However, to fully harness the potential of this technology we need to provide software developers with the proper abstractions and tools. To this end we have developed a series of programming systems to raise the level of abstraction in wireless sensor network programming.

During 2009, a team of researchers from the University of Trento and the Bruno Kessler Foundation deployed a network of autonomous tiny wireless sensors (see Figure 1) to monitor the structural integrity of Torre Aquila, a 31 metre tall medieval tower that forms part of the Buonconsiglio castle in downtown Trento (see Figure 2). The tower is a heritage treasure: it contains a series of medieval frescoes inter-

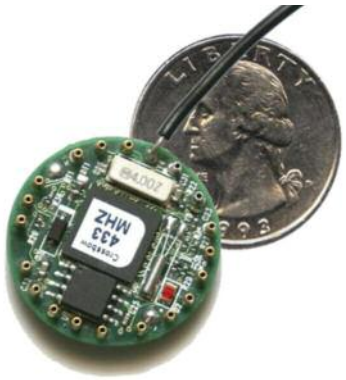


Figure 1: wireless sensor

nationally renowned as one of the few examples of non-religious medieval paintings in Europe. The main motivation for the deployment was to assess the condition of the tower in light of plans to construct a road tunnel nearby, which could have put at risk the tower's integrity.

Among the many insights provided, deformations readings revealed a phenomenon known to structural engineers as "the breath of the structure". The wireless sensors showed one of the tower's facades to elongate during the day, when the sun hits the facade, and to shorten during the night, when the temperature drops and the materials shrink accordingly. Until the wireless sensor network (WSN) was deployed, observations of this kind were either unattainable or required invasive monitoring technologies that are simply not feasible in a heritage building.

Torre Aquila is just one example of the applications enabled by Internet of Things technologies. These promise to blend today's Internet, whose modes of use and operation are mostly limited to the digital domain, with the physical world at unprecedented levels of detail. At the interface between these two worlds we find technologies such as smartphones, wireless sensor networks, and RFIDs, enabling a range of diverse applications in domains such as environmental monitoring, health-care, and factory automation.

Owing largely to small form factors, batter-powered operation, and cost considerations, Internet of Things devices are often severely resource constrained compared to today's laptops and desktop PCs. A typical wireless sensor network device, for instance, offers about the same computing and communication power as an early 80s' PC. This generates a plethora of research and technical challenges, spanning from hardware design to routing protocols and distributed algorithms.

Out of all these challenges, providing proper programming abstractions and tools is often deemed to be a key problem. These are the means whereby domain experts and software developers will be able to fully harness the potential of Internet of Things technologies. Unfortunately, the current state of the art falls short of expectations. In particular, most WSN real-world applications are designed and implemented right atop the operating system facilities, resembling the early days of distributed computing when the C language and network sockets were de facto standard programming tools. Such an approach likely distracts developers from the application goals by forcing them to deal with many low-level details. As a result, the resulting implementations have often performed below expectation.

To remedy this situation, the research community has developed a series of programming systems trying to raise the level of abstraction in WSN programming. Nevertheless, developers have hitherto rarely used these systems in real deployments. This is a consequence of having many overly

Figure 2: Torre Aquila

complex solutions designed with little attention to real-world issues. Developers, frustrated by the difficulty of understanding the system operation underlying the language constructs, then resorted to C-like languages, ultimately arguing that higher-level abstractions were not feasible in real-world deployments due to resource scarcity.

With Torre Aquila, in contrast, we demonstrated that higher-level WSN programming abstractions are not only feasible in real-world deployments and do increase the programming productivity, but they are actually necessary to realize complex applications. Indeed, we designed and implemented the monitoring system in Torre Aquila atop our TeenyLIME middleware instead of the operating system. TeenyLIME abstracts away the fragmentation of memory space across neighbouring devices under a single memory space. Such design allowed us to push more functionality in the memory-scarce WSN nodes that we would have been able to do with the operating system alone. Our abstractions allowed many mechanisms across different functionality to be factored out, ultimately resulting in smaller overall memory occupancy.

Torre Aquila and the TeenyLIME middleware are not isolated examples. As much as we conceived further abstractions to tackle different programming challenges (eg the Logical Neighborhood abstraction and the Squirrel system), we also built real-world applications with them, notably including safety-critical ones like closed-loop control in operational road tunnels. The resulting systems performed effectively and efficiently, providing fine-grained environmental data or efficient control in a range of situations.

Of course, many challenges still lie ahead in this and closely related fields. The programming challenge itself is far from being solved. We will be in the position to claim so, for example, when domain experts will develop Internet of Things applications with little or no knowledge of embedded systems and distributed programming. We believe however, that one of the grand challenges on the horizon will be the testing and verification of Internet of Things applications, especially prior to deployment. Devising effective solutions in this field will increase confidence in Internet of Things technology and thus the opportunities to investigate novel applications, creating a virtuous circle that will ignite further developments, in the same way that it happened for the "standard" Internet.

Acknowledgements: The research described here is the result of joint efforts of the author with many talented researchers at Politecnico di Milano, University of Trento, Bruno Kessler Foundation, and SICS, the Swedish Institute of Computer Science. The author is deeply indebted to all of them.

Luca Mottola is a winner of the 2011 ERCIM Cor Baayen award.

Links:
http://www.sics.se/~luca
htp://www.sics.se/nes

Please contact:
Luca Mottola, SICS, Sweden
E-mail: luca@sics.se

DIAMONDS do IT with MODELS: Innovative Security Testing Approaches

by Ina Schieferdecker, Axel Rennoch and Jürgen Großmann

Although security and model-based testing are not new areas of research, they are still under development and highly relevant. In particular, their combination is a challenge for academic work and industrial applications. Some examples of systematic and automated security testing include: security functional testing, model-based fuzzing, risk-oriented testing and the usage of security test pattern.

Multiple standardization committees provide significant efforts in the context of security testing. They cover fundamental frameworks but also detailed test specifications for concrete technologies. The range of activities is very large and includes classical concepts from security evaluation using Common Criteria (ISO/IEC 15408) for Information Technology Security Evaluation (CC) and innovative European activities from ETSI like TVRA (Threat Vulnerability Risk Analysis).

The CC is an international standard for the certification of IT-security products. The evaluation process is largely driven by developer documentation and focuses on product development, security testing and vulnerability assessment. In addition to creating trust in the product's quality, the evaluation results also allow the customer to compare the security functionality of similar products.

The TVRA method developed by ETSI benefits from the CC's work by using a well-established domain-independent generic catalogue of security functional requirements (SFRs), and is characterized by the following concepts and approaches: Threat types like interception, manipulation, denial-of-service, repudiation of messages; security objectives like confidentiality, integrity, availability, authenticity, accountability; UML to model relationships within systems; methods to analyze/evaluate threats, risks, vulnerabilities; calculation of attack potential; and the security requirements taxonomy for SFRs (from CC).

CC and TVRA both require further knowledge about how to derive security tests from the TOE description. The ITEA DIAMONDS project invests in the automatic generation of security tests from system models. The security tests needed in the quality assurance phase and/or evaluation procedures may be derived manually from the system description. Model-based approaches are currently used to contribute towards the generation of security tests. DIAMONDS work includes fuzzing, risk-based testing and security test pattern.

The aim of fuzzing is to find deviations of the real System under Test (SUT) to its specification that lead to vulnerabilities because invalid input is processed by the SUT instead of being rejected. Such deviations may lead to undefined states

of the SUT which can be exploited by an attacker, for example by allowing a denial-of-service attack because the SUT is crashing or hanging.

Risk-based security testing can generally be introduced with two different goals in mind. On the one hand, risk based testing approaches can help to optimize the overall test process. The results of the risk analysis, ie the results of threat and vulnerability analysis, are used to guide the test identification and may complement requirements engineering results with systematic information concerning threats and vulnerabilities of a system. A comprehensive risk assessment additionally introduces the notion of risk values, that is the estimation of probabilities and consequences for certain threat scenarios.

The "Security Testing chocolate box" presented by DIAMONDS at the ITEA2/ARTEMIS co-summit event 2011 in Helsinki provides a catchy example and a basic model of most security relevant terms (using CORAS) in the context of model-based security testing. It uses a chocolate box as an analogy of a SUT / Target of Evaluation (TOE), different types of chocolate as the assets, and Fredi, an intelligent fox, as the threat that wants chocolate. A security analysis focuses on the box interfaces, vulnerabilities (cover plate and a slot in the cover plate) and threat scenarios (using some tools to get the assets out of the box), in order to avoid "unwanted incidents", ie the chocolate leaving the box.

Details of fuzzing and risk-based testing can be found in the DIAMONDS deliverables. The project also looks at approaches for capturing security test patterns to create an initial repository thereof, based on the weaknesses and strengths of the systems involved. Like other types of pattern, security test patterns will be both procedural and structural, leading to various sorts of test artefacts, ranging from high-level test activities and methods, via more specific test requirements through to concrete elements of test design, eg test architecture, test behaviour and test data.

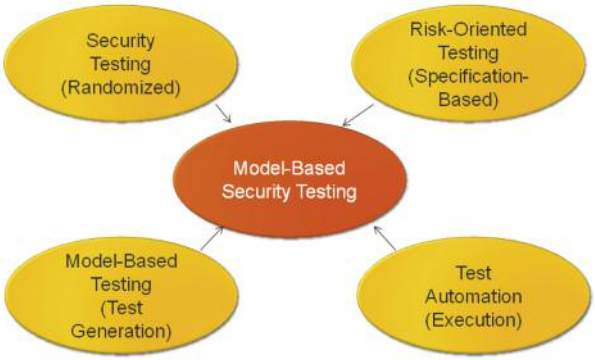


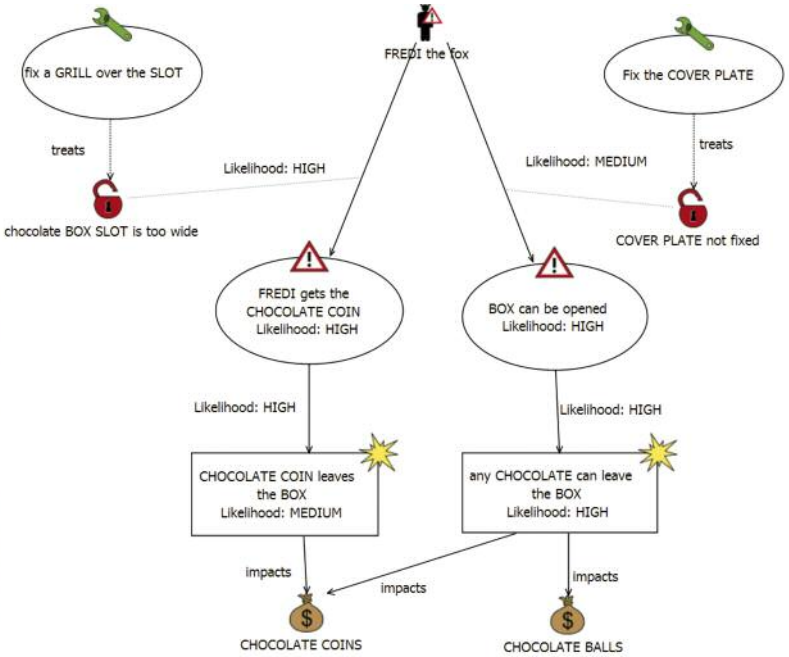
Figure 1: Security testing combined approaches

In the ITEA DIAMONDS project, 22 partners from six European countries are involved in case studies from the following domains: Banking, Smart Cards, Industrial Automation, Radio Protocols, Transport/Automotive, and critical infrastructures. Based on votes cast by participants at the ITEA2/ARTEMIS co-summit 2011 the DIAMONDS project won the prize for the best and most understandable ITEA2 project. We thank our colleagues from the project team at FOKUS and the international project consortium for their support.

Link:
http://www.itea2-diamonds.org

Please contact:
Ina Schieferdecker
Fraunhofer FOKUS, Germany
Tel: +49 30 3463 7241
E-mail: ina.schieferdecker@fokus.fraunhofer.de

(a) DIAMONDS chocolate box



(b) CORAS model

Figure 2: Security Models

dtk - A Metaplatform for Scientific Software Development

by Julien Wintz, Thibaud Kloczko, Nicolas Niclausse and David Rey

dtk is a metaplatform for software development, providing the foundations needed to develop dedicated modular scientific platforms. It aggregates functionalities embedded using low-level and interchangeable software entities - plugins - and orchestrated through high-level software entities - scripts, compositions or user interface elements. It then overcomes recurring aspects of advanced software development cycles to enable research teams to focus on research code.

This platform provides an abstraction for each of three major concepts - data, algorithm and view - all common to any scientific domain. Thus, each specific research team or group of teams can specialize these concepts within their own research area. To this end, dtk implements a mechanism for aggregating these specializations through plugins. Moreover, numerous peripheral development layers are available such as high level wrapping and scripting, visual programming, distributed computing and immersive visualization.

This platform is non-invasive in terms of codes developed by either third party communities or research teams and acts as a link between software components.
In addition, dtk supplies several tools for engineers or researchers to easily prototype experiments: integrated development environment, generators, tutorials, examples and integrated documentation system.

Below is a brief overview of three platforms made using dtk, in various fields of scientific research, namely, medical imaging, computational fluid dynamics and algebraic geometry. Sharing the same foundations, these software platforms not only aggregate code within the same application field, but also allow plugins to be interchangeable, so that bleeding edge experimentations combining habits and algorithms of very distinct scientific communities become possible.

MedInria

MedInria is medical image processing and visualization software. Through an intuitive user interface, it offers both standard and innovative functionalities for medical images such as 2D/3D/4D image visualization, registration, segmentation or tractography. It mainly uses core concepts of dtk: data, processes and views, as well as helper concepts such as readers, writers, converters, interactors etc.

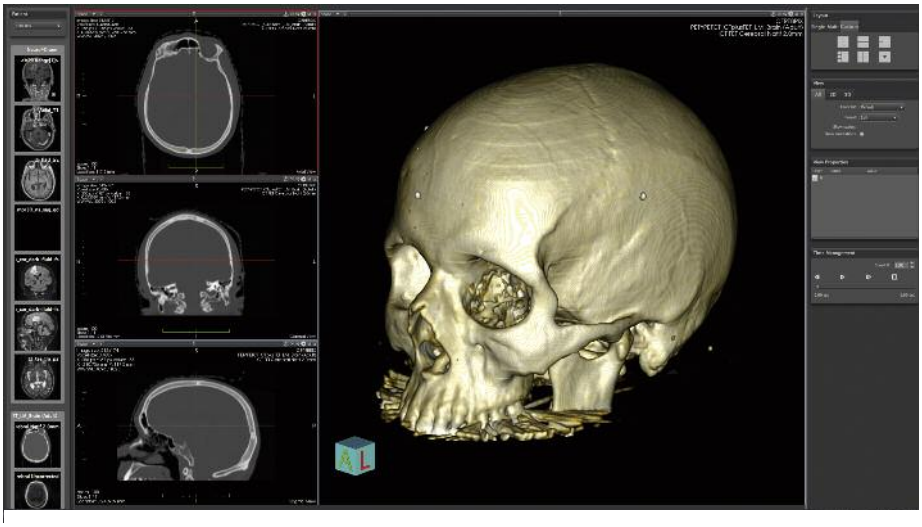


Figure 1: Medinria: a medical imaging platform

It is also maintained as an application demonstrating the popularization of immersive visualization and interaction technologies, using the virtual reality layer of dtk: users can visualize medical images in stereo and interact with them manually, in a virtual reality centre.

MedInria comes in two flavors: a free academic platform developed by four Inria teams (Asclepios, Athena, Parietal and Visages) and an industrial platform aiming at transferring research tools to hospital.

num3sis

num3sis is a modular platform devoted to scientific computing and numerical simulation. It is designed to handle complex multidisciplinary simulations involving several fields such as Computational Fluid Dynamics (CFD), Computational Structural Mechanic (CSM) and Computational Electro-Magnetics (CEM). In this context, the platform provides a comprehensive framework for engineers and researchers that speeds up implementation of new models and algorithms.

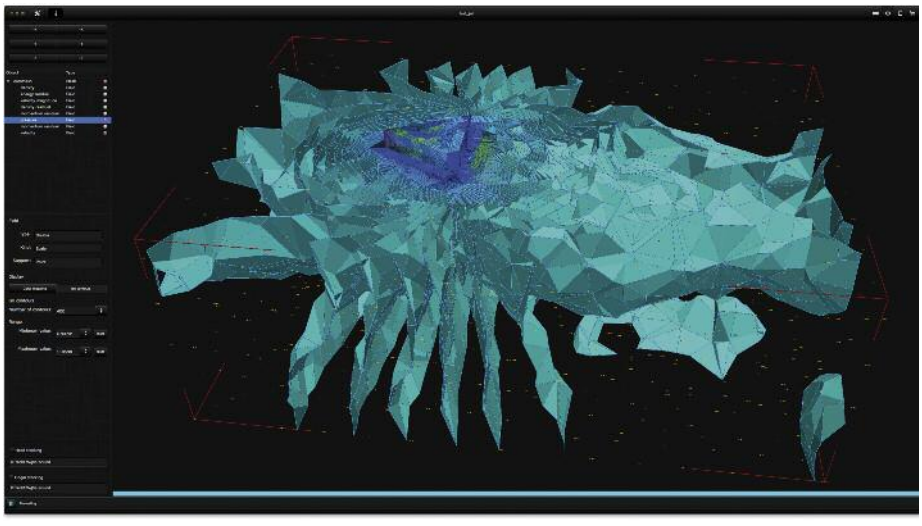


Figure 2: Num3sis: a multi disciplinary numerical computing platform

From a software engineering point of view, num3sis specializes and extends

Figure 3: Axel: an algebraic geometric modeler

some layers of the meta-platform dtk, especially its core and composition layers. The core layer enables users to define generic concepts for numerical simulation such as mesh or finite-volume schemes which are then implemented through a set of plugins. The composition layer provides a visual programming framework that wraps these concepts inside graphical items - nodes. These nodes can then be connected to each other to define data flows (or compositions) corresponding to the solution of scientific problems.

num3sis provides a highly flexible, re-usable and efficient approach to develop new computational scenarios and takes advantage of existing tools.

axel

Whilst most Computer Aided Design (CAD) tools or geometric modelers hide internal representations of objects and focus on design, axel places their representation in the heart of the modeling process.

Extending core concepts of dtk, axel provides elementary geometric entities such as curves, surfaces or volumes, and lets advanced users interact with their representations. These representations may vary from discrete to continuous (including parametric and implicit).

As manipulating such fundamental mathematical representations, with cutting edge research tools, requires some connections with a Computer Algebra System (CAS) (in this case, Mathemagix), axel takes advantage of the script layer of dtk to wrap its concepts in various scripting languages.

Today, dtk has met its primary goal: it acts as a platform enabling researchers and engineers to develop platforms, more easily, quickly and effectively, by focusing only on research code within smaller software entities - plugins -, the cohesion being assured by dtk. Furthermore, the underlying software architecture makes it possible to keep a real case-by-case licensing policy.

dtk, as a software platform, also proposes and maintains a connection with hardware platforms of research centres, such as high performance computing centres or virtual reality centres. Another focus is keeping abreast of technological evolutions. In this context, new platforms such as mobile platforms are investigated.

Finally, such a technical structure for software development results in great human organization, where it is possible to promote new development paradigms, creating connections between developers, not only within a project, but also across all dtk based projects.

Links:
http://dtk.inria.fr
http://dtk.inria.fr/guides
http://dtk.inria.fr/blog
http://med.inria.fr
http://num3sis.inria.fr
http://num3sis.inria.fr/blog
http://axel.inria.fr

Please contact:

for dtk:
David Rey, Julien Wintz, Inria, France,
E-mail: david.rey@inria.fr, julien.wintz@inria.fr

for Medinria:
Olivier Commovick, Inria, France, Visages project-team
E-mail: olivier.clatz@inria.fr
Olivier Clatz, Inria, France, Asclepios project-team
E-mail: olivier.commovick@inria.fr,

for num3Sis:
Régis Duvigneau, Inria, France, Opale project-team
E-mail: regis.duvigneau@inria.fr
Stéphane Lanteri, Inria, France, Nachos project-team
E-mail: stephane.lanteri@inria.fr

for axel:
Bernard Mourrain, Inria, France, Galaad project-team
E-mail: bernard.mourrain@inria.fr

CloudNets: Combining Clouds with Networking

by Anja Feldmann, Gregor Schaffrath and Stefan Schmid

In the future, Internet Service Providers (ISP) may offer on-demand, flexible virtual networks connecting different locations and heterogeneous cloud resources with Quality of Service (QoS) connectivity guarantees (such as maximal latency or minimal bandwidth).

The virtualization paradigm is arguably the main motor for networking innovation in the future. Virtualization decouples services from the underlying infrastructure and allows for resource sharing while ensuring performance guarantees. Server virtualization (also known as node virtualization) has already been a huge success and is widely used, for example, in the clouds.

However, cloud virtualization alone is often meaningless without taking into account the network needed to access the cloud resources and data. Thus, to provide deterministic performance guarantees, cloud virtualization needs to be extended to the access and communication network.

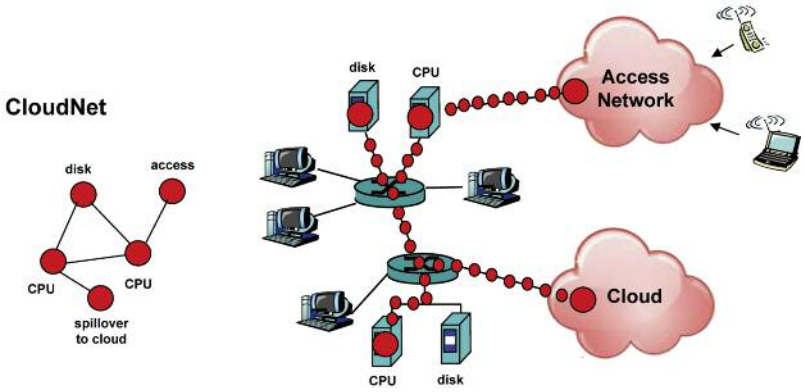


Figure 1: CloudNet infrasctrucure and embedding

"CloudNets" take the virtualization paradigm one step further and offer such a unified approach. A CloudNet describes a virtual network topology where the virtual nodes represent cloud resources (eg storage or computation) which are connected by virtual links.

We envision that in the near future, flexibly specified CloudNets (eg used for multi-media conferencing, gaming, social networking, or bulk data transfer) can be requested and realized at short notice and for a desired period of time. For example, a CloudNet may specify geographical constraints (eg realization only in clouds at distance less than 100m), topological constraints (eg some virtual links are half-duplex, full-duplex or even describe a shared medium), capacity constraints (eg minimal reserved storage or bandwidth), performance constraints (eg all users in Germany can access an application with a maxumum delay of 50ms), compatibility constraints (eg some nodes must be binary compatible), or constraints on how the resources required by the virtual node may be split among multiple physical nodes (eg to aggregate resources or ensure redundancy).

There are many applications for CloudNets. In a multi-tenant production data centre or cloud context, the isolation and QoS networking properties of CloudNets are attractive to ensure that jobs do not miss hard deadlines due to unpredictable changes in the load; this guarantees application performance and avoids resource inefficiencies that eventually lead to provider revenue losses.

CloudNets can also be used to seamlessly connect geographically separated clouds or nano data centres, aggregating a huge amount of resources. Another interesting use case is flexible out-sourcing or cloud bursting: a corporate infrastructure or an in-house data centre is connected to public clouds, and at times of high demand, certain applications are migrated to the cloud.

In many of these scenarios, it is unlikely that a CloudNet request specifies every detail: for instance, in our outsourcing scenario, no specific cloud provider may be named explicitly, and a computational CloudNet request or a CloudNet for delay-tolerant bulk data transfers may specify a flexible time window for the realization. This flexibility in the specification can be exploited for optimizations, for instance to choose the cheapest cloud provider, or to choose the realization period where the load on the infrastructure is low or where electricity is cheap.

Within the limitations of the specification, a CloudNet can also be migrated. A CloudNet provider can use migration to co-locate CloudNets in times of low demand, eg to save energy if the remaining network components can be switched off, or to perform maintenance work. Migration can also be used to improve the Quality-of-Experience: for instance, an SAP server, a gaming server or even (parts of) a CDN server can adaptively migrate closer to the location of the users which reduces the latency. In a global application, the CloudNet servers will cycle around the earth, whereas in a more local application, the servers will follow the commuters downtown in the morning and back to the suburbs in the evening; at night, the virtual servers may switch to a different technology or even be shut down completely.

The concept of CloudNets is particularly interesting for ISPs. The possibility to offer new innovative services may increase revenues, and the more efficient usage of the given resources and the simplified network management can reduce the investment costs for new technology and decrease operational costs. Moreover, ISPs have the competitive advantage of knowing not only the network infrastructure in detail but also the current demand. This allows for various optimizations that could not be performed to a comparable extent by CDN providers for example. The explicit knowledge of the customers' desired specifications (inferrable from the CloudNet requests) can also simplify the network provisioning and allow the ISP to assess the cost and impact of reconfigurations.

At the same time, note that CloudNets may have a large geographic footprint and cannot be instantiated unilaterally by a

single ISP, but require cooperation. This can lead to new business roles. For instance, virtual network providers may emerge which interact with several infrastructure providers as resource brokers or resellers. Such a virtual network provider is not necessarily an independent entity, but may just as well constitute a new subunit inside an existing ISP.

Although many algorithmic and economic implications are not yet well-understood, we believe that the virtualization technology (eg VMWare for server virtualization, or VLANs and OpenFlow for link virtualization) is ripe to realize the vision of CloudNets.

Link: http://www.inet.tu-berlin.de/

Please contact:
Anja Feldmann, Gregor Schaffrath, Stefan Schmid
TU Berlin & Telekom Innovation Laboratories (T-Labs), Berlin
E-mail: {anja,grsch,stefan}@net.t-labs.tu-berlin.de

Innovation in Disaster Management: Report from Exercise EU POSEIDON 2011

by Catherine E. Chronaki, Vasilis Kontoyiannis, Panayiotis Argyropaidas, and Dimitris Vourvahakis

Innovative ICT services linking disaster reports, live e-Triage data and SMS/Twitter alerts to maps for situational awareness were part of EU POSEIDON 2011, a large-scale European civil protection exercise held in Crete on October 24-26, 2011.

Exercise EU POSEIDON 2011 marked 10 years of the European Civil Protection Mechanism (ECP). It ran for two days in real-time and involved four levels of civil protection (local, regional, national, European). More than 300 participants attended representing fire brigade, Emergency Medical Services (EMS), health authority, port authority, police, municipalities, power plant, volunteers, along with search and rescue teams from Greece (Red Cross), France (PCSF), and Cyprus (Civil Defense). The Exercise Command comprised members of the Regional Directorate of Civil Protection and the General Secretariat of Civil Protection, Hellenic Ministry of Citizen Protection. Observers from country-members of the ECP attended the exercise.

The exercise was organized in the context of the POSEIDON project "Earthquake followed by Tsunami in the Mediterranean Sea" co-funded by DG ECHO, and provided a unique opportunity to validate new applications for early warning and communications systems as well as procedures by which to inform the public of emergency measures to be undertaken in a disaster scenario based on the tsunami of 365a.d. (Flouri et al., ERCIM News 81).

SMS/Twitter messages for public awareness alerts/warnings were evaluated in terms of understandability, credibility, usability and usefulness by exercise participants. SMS in different languages included URLs to information resources, risk and resource location maps, as well as alert messages in the Common Alerting Protocol (CAP ITU-T X.130). CAP messages reported on the pending tsunami and the status of the disaster in an interoperable format also used by GDACS. Similar messages on Twitter allowed team leaders, observers, and media to follow the progress of the exercise from their smartphones delivering a high-level log of the exercise.

Following a disaster, time is the most critical resource in the management of emergencies. There are large numbers of unidentified victims or people missing. At the Coordination Centres, timely information is needed to assess the situation, weigh options and engage the limited means and resources in

During the civil protection exercise

the most effective manner. In the field prompt triage of victims (<30 seconds per victim), seamless identification and tracking from rescue to hospital or shelter, are of paramount importance.

e-Triage, an innovative technology developed by FORTH-ICS in collaboration with EMS-Crete combined the START protocol on smartphones held by rescuers with colour-coded triage cards with QR codes, to meet these requirements. In EU POSEIDON 2011, e-Triage was used by rescuers on realistic cases involving more than 70 victims played by volunteers in Chania and Heraklion.

Victims were located by rescue dogs and retrieved by firefighters and rescue teams. Trained rescuers performed protocol-based e-Triage and provided the victims with a bracelet marked with a unique QR code, a "Green", "Yellow", "Red", or "Black" sticker and a letter indicating the affected body system. According to the START protocol, "Green" victims

were transferred and checked-in to the "Lightly Wounded-Walking Area" managed by the Hellenic Read Cross, "Black" victims were transferred and checked-in to the "Dead or Expected Area" managed by the Hellenic Police, while "Red" and "Yellow" victims were checked in the on-site "Field hospital" managed by the EMS. The medical personnel in the Field Hospital carried out 2nd level triage in 3-5 minutes, ie collected additional information on the victims and performed medical intervention as needed. At the same time, they placed requests for ambulances and dispatched victims based on severity.

The Local Coordination Centre at the disaster area was in contact with the EMS Headquarters to report regularly on the unfolding situation. With e-Triage there was immediate and transparent sharing of information regarding the number, type, and gravity of the victims to the EMS Coordination Centre, the hospital, and the regional Coordination Centre. At the regional Coordination Centre a map showed different disaster areas with running counts of victims, resources, and needs. In this way, the needs of the different sectors (EMS Coordination Centre, Disaster areas, Hospital emergency wards, evacuation centres, etc.) were effectively communicated supporting prompt and effective decision making.

The technical infrastructure supporting e-Triage was a local WiFi network connected to the EMS Coordination Centre by secure GSM or satellite network connectivity. Disconnected operation was supported by local Web and Database servers. Thus, the disaster area simulated a WiFi bubble resilient to possible fluctuations or loss of Internet connectivity for the eTriage applications on netbooks, tablets or smartphones. If Internet connectivity was present, replication of the local database to the EMS database was performed at preset intervals (three minutes). Skype communication from the local coordination centre, the different areas, and the EMS coordination centre was also used if Internet was available.

In the absence of Internet connectivity, VHF radio was used to report a summary of the situation to the EMS coordination centre including number of victims in different areas. As soon as Internet connectivity was restored, the applications were resynchronized. Thus, e-Triage proved very effective in supporting coordination and cooperation, bridging information gaps along the chain of emergency management, offering an innovative approach to the very demanding setting of disaster management.

Links:
European Civil Protection:
http://ec.europa.eu/environment/civil/index.htm
START Triage: http://asimplematter.com/CMH/page10.html
QR Codes: http://www.qrstuff.com/qr_phone_software.html
GDACS: http://www.gdacs.org/index.asp

Please contact:
Catherine Chronaki, FORTH-ICS, Greece
E-mail: chronaki@ics.forth.gr

Media Search Cluster White Paper on "Search Computing"

by Yiannis Kompatsiaris, Spiros Nikolopoulos, Thomas Lidy and Andreas Rauber

A white paper on "Search Computing: Business Areas, Research and Socio-Economic Challenges" was recently published by CHORUS+, a European Coordination Action on Audio-Visual Media Search. It provides an overview of the business areas, the research challenges and the socio-economic aspects related to "Search Computing". It consolidates the experts' opinion about the future opportunities and trends in the search industry.

Search has become an important component of many diverse ICT applications. A large number of business and application areas such as: a) the Web, b) mobile devices and applications, c) social networks and social media, and d) enterprise data access and organization, depend on the efficiency and availability of search techniques being able to process and retrieve heterogeneous and dispersed data. Such techniques of "Search Computing" are directly related to a number of research topics ranging from multimodal analysis and indexing to affective computing and human aspects as well as to various socio-economic challenges including business models, open innovation, legal and ethical issues.

The objective of the European Coordination Action CHORUS+ is to coordinate national and international projects and initiatives in the audio-visual search domain and to extend this coordination to non-European countries through mutual information exchange, dissemination and cross fertilisation. CHORUS+ provides a collaborative platform to support the efforts and foster collaboration between researchers as well as industry. The platform is an open repository and community effort to provide an overview of resources in the field of audio-visual search: available Tools, Datasets, and Publications as well as Benchmarking Events. At the business level, CHORUS+ aims to foster discussion and avoid fragmentation.

CHORUS+ activities include conferences, workshops and specialized Think-Tank events to discuss emerging technologies within a group of representatives from industry and academia. Based on the resulting findings, CHORUS+ publishes reports and white papers addressing technology producers, content owners and consumers. Recently, CHORUS+ together with 23 contributors from 12 R&D projects of the FP7 Media Search Cluster has published a White Paper on "Search Computing: Business Areas, Research and Socio-Economic Challenges" providing an overview of the business areas, the research challenges and the socio-economic aspects related to "Search Computing".

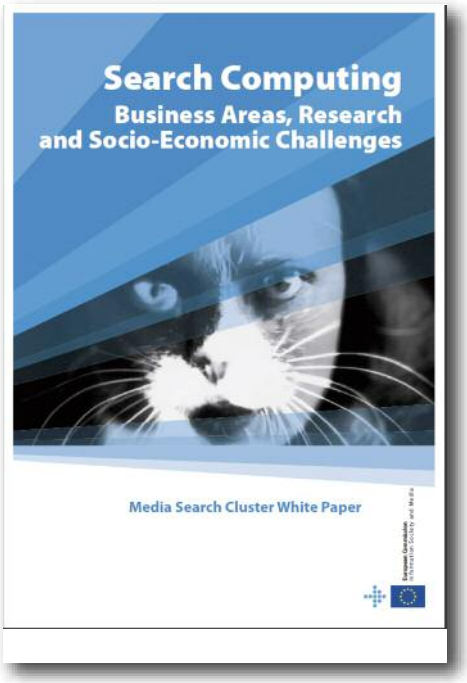
The business areas include mobile, enterprise, social networks and music, focusing on current status, technology and application requirements and the arising new needs. Specific technologies and challenges are identified such as device limitations and location-based opportunities in mobile search

but also challenges that are applicable across domains such as limitations of text-based approaches and new directions for content-based search. Despite significant research, there are still unsolved problems. The White Paper covers:

- Multimodal search enabling queries and interaction independently of the form of the content
- Affective-based search taking into account both the user's emotional state and the sentiment contained in multimedia documents
- Event-based representation and analysis as an efficient and user-centric approach for the annotation and retrieval of content
- User experience aspects including new generations of search interfaces (eg, visual search interfaces, augmented reality, 3D browsing in virtual places) to enable novel forms of search input and to avoid losing the overview over the incredible amount of content
- Large-scale indexing addressing the scalability issues related to the huge volume of content available on the Web by using social connections to implement targeted and thus smarter indexes
- Content-aware network nodes extending search in the network with challenges related rather to "discovery at the network" than "searching in the search engine"
- Real-time approaches and architectures enabling to push information to users as fast as it is available, and maintaining the balance between quality, authority, relevance and timeliness
- Content diversity in knowledge, providing the ability to identify and exploit the aspects that differentiate a piece of information from another
- Aggregation, mining and Linked Open Data enabling data collection and aggregation from social networks and interlinking data chunks under the Linked Open Data principles
- Standardization initiatives enabling a complete set of new technologies to face the market and achieve a deep impact on future business.

The socio-economic challenges of "Search Computing" are analysed next including business models, search and open innovation methodologies, benchmarking, legal and ethical issues (data protection and user privacy). New business models are needed in applications where user-generated content plays an important role in order to allow both commercial exploitation and protect user rights. Open innovation is the use of purposive inflows and outflows of knowledge to accelerate internal innovation and expand the markets for external use of innovation. Benchmarks are valued for their ability to streamline research by eliminating redundancy, enabling direct performance comparison between algorithms, increasing efficiency by sharing resources between research sites and providing a concrete framework in which researchers interact in a productive mixture of competition and collaboration. Finally, social networks pose new challenges to legal and ethical issues that should be carefully considered for proper usage and processing of personal data.

The overall conclusion of the White Paper is that "Search Computing" is an active area, with technologies needed in a number of markets and applications. Many issues are still unsolved and new ones have appeared resulting in many of technological and socio-economic research challenges.



Cover of Media Search Cluster White Paper

Links:
The full White Paper is available from:
http://avmediasearch.eu/Search_Computing_white_paper
CHORUS+ Website: http://avmediasearch.eu
CHORUS+ Collaborative Platform:
http://avmediasearch.eu/wiki/

Please contact:
Yiannis (Ioannis) Kompatsiaris
Informatics and Telematics Institute- Centre for Research and Technology Hellas, Greece
E-mail: ikom@iti.gr

Andreas Rauber
TU Wien (AARIT), Austria
E-mail: rauber@ifs.tuwien.ac.at

Models and Logics for Quantitative Analysis

by Flemming Nielson
and Ender Yüksel

The third annual meeting of the ERCIM Working Group on Models and Logics for Quantitative Analysis (MLQA) took place on Monday 5 September 2011 as part of the 22nd International Conference on Concurrency Theory (CONCUR 2011) organized by the RWTH Aachen University in Germany. More than twenty researchers, from senior researchers to PhD students, attended this meeting.

The theme of the meeting was the role of continuous modeling and analysis techniques in computer science and applications. Traditionally, continuous domains have been used systematically in quantitative modeling and analysis, in the form of time, probability, etc. The focus of the MLQA meeting, instead, was on issues like continuous approximations of population sizes, ordinary differential equations (ODEs), semantics of stochastic process calculi or other languages, and ODE limits of Markov Chains.

The first talk, by Professor Jean-Yves Le Boudec, explained the meaning of basic concepts such as "mean field approximation", "fluid approximation", "fixed point method" and "decoupling assumption". Relating these concepts to ODEs and explaining how fast simulation methods can be derived, Professor Le Boudec showed that mean field approximation and fluid approximation are generally valid whereas the fixed point method and the decoupling assumption require more care and may not hold even in simple cases. This talk was followed by two talks that focused on the relationship between ODEs and stochastic process algebras, by Professor Jane Hillston and Dr. Luca Bortolussi. Professor Hillston discussed the recent work on population-oriented models described in stochastic process algebras and interpreted as a set of ordinary differential equations, the so-called fluid approximation. Dr. Bortolussi concentrated on the mathematical foundations of the fluid-approximation of stochastic process algebra models and discussed various related issues including extensions of the limit theorem.

Professor Jane Hillston giving a talk on stochastic process algebras and ordinary differential equations

An application of fluid approximations is the analysis of Markov population models. Dr. Verena Wolf discussed the stochastic hybrid analysis of Markov population models in her presentation. Dr. Wolf explained various numerical techniques to analyze such models over time and in equilibrium. She also talked about the importance of stability analysis for Markov population models and presented techniques to reason about the mathematical foundations of multi-stability and oscillation of stochastic models.

In her talk on the approximation of continuous space systems and associated metrics and logics, Professor José Desharnais surveyed approximation techniques for Labelled Markov Processes - a model of probabilistic processes where the state space is a measure space. Professor Desharnais showed that they obtained a practical approximation by aggregating states that share some similarities.

Professor Flemming Nielson convened a small thematic working group on Hidden Markov Models. The aim of this initiative was to identify researchers that would be interested in developing logics, query languages, and model checkers that would be able to deal with some of the applications of Hidden Markov Models. The efforts in this area would hopefully lead to useful tools in much the same way that existing stochastic model checkers perform the analysis of Markov Processes.

In a final business meeting a steering committee was formed for planning the upcoming activities of the working group. The steering committee is comprised of Flemming Nielson (Technical University of Denmark), Diego Latella (ISTI-CNR), Jane Hillston (University of Edinburgh), Herbert Wiklicky (Imperial College London), and Erik de Vink (Eindhoven University of Technology).

For more information please consult the MLQA wiki where most of the presentations are available and where details of the next meeting in 2012 will be posted as well as the plans for further stimulating interaction among members of the working group.

Link:
http://wiki.ercim.eu/wg/MLQA

Please contact:
Flemming Nielson
ERCIM MLQA Working Group Coordinator
DTU (Technical University of Denmark) Informatics, Denmark
Tel: +45 45 25 37 35
E-mail: nielson@imm.dtu.dk

Concentration and attention during the talks

ERCIM/EWICS/DECOS Dependable Cyber-physical Systems Workshop at SAFECOMP 2011

by Erwin Schoitsch
and Amund Skavhaug

The annual ERCIM/EWICS DECOS Dependable Cyber-physical Systems Workshop (formerly "Dependable Embedded Systems Workshop") was held in Naples, Italy on 22 September 2011 in conjunction with SAFECOMP 2011. The workshop was organized by the ERCIM Dependable Embedded Systems Working Group, together with EWICS TC7 (European Workshop on Industrial Computer Systems Technical Committee 7) and the DECOS Interest Group (DIG) "Dependable Embedded Components and Systems, an IP in FP6".

The theme of SAFECOMP 2011 was "Safety and Security of Computer-based Systems and Infrastructures: from Risk Assessment to Threat Mitigation". About 85 participants listened to interesting talks in eleven sessions and a poster session. Topics of the sessions were RAM Evaluation I and II, Complex Systems Dependability I and II, Formal Verification I, II and III, Risk and Hazard Analysis, Cybersecurity, Case Studies, Optimization Methods. Key Note speakers were:

- Paulo Verissimo from University of Lisboa (Portugal): "Security and Dependability Risks of Critical Information Infrastructures (or why Bang! is different from Crash)."
- Gerard J. Holzmann from Caltech and NASA Jet Propulsion Lab (USA): "Software Safety and Software Complexity"
- Andrea Bondavalli from University of Firenze (Italy): "Model based resilience assessment of critical information infrastructures".

The last day of SAFECOMP was dedicated to two tutorials and two workshops which were held simultaneously. The full-day ERCIM/EWICS/DECOS workshop attracted 25 participants. According to its broad scope, it comprised three topics in four sessions:

- Dependable and resilient embedded systems I and II
- System Safety, Systems-of-Systems
- Autonomous Systems and Robotics.

In the introductory talk, Erwin Schoitsch gave an short overview on ERCIM, EWICS and the ARTEMIS Joint Technology Initiative and Joint Undertaking, focussing on three ARTEMIS projects related to the workshop session topics: MBAT (Model-Based Analysis and Testing of Embedded Systems), SafeCer (Safety Certification of Software-Intensive Systems with Reusable Components) and R3-COP (Resilient Reasoning Robotic Co-operating Systems).

In the first two sessions, "work in progress", it was reported from several ARTEMIS projects :

- CESAR, the largest ARTEMIS project with 55 partners and a total budget of €68 million, aims to create an ARTEMIS Reference Technology Platform for safety-critical embedded systems. CESAR was presented by Roland Mader, AVL company, Austria);
- ACROSS (about Network on Chip (NoC), time-triggered Ethernet based), presented by Oliver Höftberger from company TTTech, Austria)
- pShield (on SPD – Secure, Private, Dependable Power Node Embedded System), presented by Przemysław Osocha, SESM company, Italy.

These presentations were accompanied by two other talks on co-modelling and co-simulation for Dependable Embedded Systems (DESTECS project, presented by Ken Pierce, Newcastle University, UK) and "What UML based mutation testing can tell us about a system (and what not)", by Rupert Schlick (AIT Austrian Institute of Technology).

The session "System Safety, Systems-of-Systems" was driven by industrial experience talks: "Distributed Safety Assessment for Airborne Systems", presented by Martin Waßmuth, EADS, Germany and "A System Approach for the Safety Demonstration of the Main Brake Pipe Recharge Inhibition Command" presented by Francesco Sperandio from d' Apollonia SpA, Italy (a consultant to ALSTOM Ferroviaria). Erwin Schoitsch concluded the session, highlighting the need for holistic system approaches: "What can we learn from regional disasters about holistic risk assessment? The systems-of-systems view of complex cyber-physical systems".

The third session was dedicated to the robotics field of research. Reports were given from R3-COP presented by Francesca Saglietti from The University of Erlangen-Nuremberg. Her presentation, entitled "Model-based Representation of Cooperative, Autonomous Systems", demonstrated the advantage of using coloured Petri nets to model complex vision- and perception based scenarios in a compact manner. Saglietti's report is complemented by a presentation available on the SAFECOMP Web site on a R3-COP market study concerning "Trends and tendencies for embedded systems in Robotics" from Antonio Feraco, company Innova, Italy.

André Dietrich from the Magdeburg Robotic Lab at the University of Magdeburg, Germany, gave a talk on a closely related issue, entitled "Model-based decoupling of perception and processing", focussing on exploiting geometric and behaviour models for improved interaction between robots and their environment, and role-based human-machine interaction.

Two presentations were about practical applications including orientation and mapping by fingerprinting methods ("Autonomous Maintenance Robot for Location Fingerprinting Methods", Janne Merilinna, VTT Finland) and "A modular software system targeted towards embedded applications exemplified by UAV usage" (presented by Amund Skavhaug (NTNU, Norway), both including nice visualization of the experimental prototypes by short videos.

The presentations led, to very lively discussions and raised a high level of interest. Overall, the workshop can be considered a success.

Link:
http://www.safecomp2011.unina.it/

Please contact:
Erwin Schoitsch
ERCIM DES Working Group chair
AIT Austrian Institute of Technology (AARIT)
Erwin.schoitsch@ait.ac.at

MediaEval 2011 Evaluation Campaign

by Gareth J. F. Jones and Martha Larson

MediaEval is an international multimedia benchmarking initiative offering innovative new tasks to the multimedia community. MediaEval 2011 featured tasks incorporating social media search, analysis of affect and location placing of images.

MediaEval is an international multimedia benchmarking initiative that offers innovative new content analysis, indexing and search tasks to the multimedia community. MediaEval focuses on social and human aspects of multimedia and strives to emphasize the 'multi' in multimedia, including the use of speech, audio, tags, users, and context, as well as visual content. MediaEval seeks to encourage novel and creative approaches to tackling these new and emerging multimedia tasks. Participation in MediaEval tasks is open to any research group who signs up. MediaEval was launched as VideoCLEF as track at CLEF 2008 and became an independent benchmarking campaign in 2010 with sponsorship by the PetaMedia Network of Excellence.

MediaEval workshop participants

MediaEval 2011 offered 6 tasks coordinated in cooperation with various research groups in Europe and elsewhere. The following tasks were offered in the 2011 season:

- Placing Task: This task required participants to assign geographical coordinates (latitude and longitude) to each of a provided set of test videos. Participants could make use of metadata and audio and visual features as well as external resources.
- Spoken Web Search Task: This task involved searching for audio content within audio content using an audio content query. It addresses the challenge of search for multiple, resource-limited languages. The application domain is the Spoken Web being developed for low-literacy communities in the developing world.
- Affect Task: This task required participants to deploy multimodal features to automatically detect portions of movies containing violent material. Violence is defined as "physical violence or accident resulting in human injury or pain". Any features automatically extracted from the video, including the subtitles, could be used by participants.
- Social Event Detection Task: This task requires participants to discover events and detect media items that are related to either a specific social event or an event-class of interest. Social events of interest were planned by people, attended by people and the social media captured by people.
- Genre Tagging Task: The task required participants to automatically assign tags to Internet videos using features derived from speech, audio, visual content or associated textual or social information. This year the task focused on labels that reflect the genre of the video.
- Rich Speech Retrieval Task: The task went beyond conventional spoken content retrieval by requiring participants to deploy spoken content and its context in order to find jump-points in an audiovisual collection of Internet video for given a set of queries.

The 2011 campaign culminated in the MediaEval 2011 workshop that was held on 1-2 September at Fossabanda Santa Croce in Pisa, Italy. Reflecting MediaEval's engagement with different research communities, the workshop was an official satellite event of Interspeech 2011. The workshop brought together the task participants to report on their findings, discuss their approaches and learn from each other. A total of 39 submissions were made to the working notes from around 35 different research sites, and almost 60 participants attended the workshop - representing a twofold increase on the 2010 workshop. In addition to organizer and participant presentations, the workshop included a practitioners' session in which projects, research sites and industry groups that are involved with MediaEval related tasks or technology presented overviews of their work and ideas. The workshop concluded with a meeting of task organizers and other interested researchers that consisted of presentations and discussions of task proposals for MediaEval 2012. The working notes proceedings for the MediaEval 2011 workshop have been published by CEUR workshop proceedings.

In addition to PetaMedia, MediaEval 2011 received support from a number of EU and national projects and other organizations including: AXES, OpenSEM, Glocal, WeKnowIt, Chorus+, Quaero, IISSCoS, Technicolor, IBM Research - India and Carnegie Mellon University.

Further details of MediaEval are available from the MediaEval website. We are now beginning preparations for MediaEval 2012. This begins with a questionnaire to the community seeking their views on proposed tasks and research questions. Feedback from the questionnaire is used to determine the research agenda for the campaign. If you are interested in receiving the questionnaire, participating in a task or even coordinating a task as part of MediaEval 2012, please contact Martha Larson.

Links:
MediaEval website: http://www.multimediaeval.org
MediaEval 2011 online proceedings: http://ceur-ws.org/Vol-807/

Please contact:
Martha Larson
Delft University of Technology, The Netherlands
E-mail: M.A.Larson@tudelft.nl

The CLEF Initiative: Conference and Labs of the Evaluation Forum

by Nicola Ferro

Since 2000, CLEF has played a successful role in stimulating research and promoting evaluation in a wide range of key areas in the information access and retrieval domain. In 2010, a radical innovation and renewal process led to the establishment of the CLEF Initiative, whose mission is to promote research, innovation, and development of information access systems with emphasis on multilingual and multimodal information by providing an infrastructure for:

- multilingual and multimodal system testing, tuning and evaluation
- investigation of the use of unstructured, semi-structured, highly-structured, and semantically enriched data in information access
- creation of reusable test collections for benchmarking
- exploration of new evaluation methodologies and innovative ways of using experimental data
- discussion of results, comparison of approaches, exchange of ideas, and transfer of knowledge.

The CLEF Initiative is structured in two main parts:

1. A series of Evaluation Labs, i.e. laboratories to conduct evaluation of information access systems and workshops to discuss and pilot innovative evaluation activities.
2. A peer-reviewed Conference on a broad range of issues, including:
   - the activities of the Evaluation Labs
   - experiments using multilingual and multimodal data; in particular, but not only, data resulting from CLEF activities
   - research in evaluation methodologies and challenges.

Due to these changes and the broader scope of the CLEF Initiative, the acronym CLEF, traditionally expanded to Cross-Language Evaluation Forum, now translates to Conference and Labs of the Evaluation Forum.

Impressions from the CLEF conference

This renewal process and the organization of the annual CLEF events are partially supported by the EU FP7 PROMISE project (Participative Research labOratory for Multimedia and Multilingual Information Systems Evaluation).

CLEF 2011: The Second Event of the CLEF Initiative

CLEF 2011 was hosted by University of Amsterdam, The Netherlands, 19-22 September 2011 as a three and a half days event where conference presentations, laboratories and workshops, and community sessions were smoothly interleaved to provide a continuous stream of discussions on the different facets of experimental evaluation.

14 papers (ten full and four short) were accepted for the Conference and published by Springer in their Lectures Notes for Computer Science (LNCS) series. Two keynote speakers highlighted important developments in the field of evaluation. Elaine Toms, University of Sheffield, focused on the role of users. She argued that evaluation has moved from an emphasis on topical relevance to an emphasis on measuring almost anything that can be quantified. Omar Alonso, from Microsoft USA, presented a framework for the use of crowdsourcing experiments in retrieval evaluation.

The community sessions at CLEF 2011 were organized around a strategic EU meeting to promote funding opportunities, a networking session on IR for scientific multimedia data organized by the Chorus Network of Excellence, an Evaluation Initiatives session with overviews from other benchmarking fora, and an infrastructure session dedicated to the DIRECT system for handling scientific data resulting from retrieval experiments.

Five benchmarking evaluations ran as labs in CLEF 2011:

- CLEF-IP: a benchmarking activity on intellectual property
- ImageCLEF: a benchmarking activity on image retrieval, focusing on the combination of textual and visual retrieval
- LogCLEF: a benchmarking activity on multilingual log file analysis, namely language identification, query classification, success of a query
- PAN: a benchmarking activity on uncovering plagiarism, authorship, and social software misuse
- QA4MRE: a benchmarking activity on the evaluation of machine reading systems through question answering and reading comprehension tests.

There were also two exploration workshops:

- CHiC (new): a workshop aimed at a systematic and large-scale evaluation of cultural heritage digital libraries

- MusicCLEF (new): a pilot lab/workshop on the evaluation of music search engines using both audio content and textual descriptions

CLEF 2012: Information Access Evaluation meets Multilinguality, Multimodality, and Visual Analytics

CLEF 2012 will be hosted by the Sapienza University of Rome, Italy, 17-20 September 2012. The Call for Lab proposals was issued at the beginning of November 2011 and 10 lab proposals and two workshops proposal were received. Seven labs and one workshops, out of which four are new, have been selected to run during 2012:

- CHiC (new): a last-year workshop turning into a benchmarking activity for the cultural heritage domain based on Europeana collections
- CLEF-IP: a benchmarking activity on intellectual property
- ImageCLEF: a benchmarking activity on image retrieval
- INEX (new): the well-known Initiative for Evaluation of XML retrieval joins efforts with CLEF to target new synergies between multilingual, multimodal and semi-structured information access
- PAN: a benchmarking activity on plagiarism detection
- QA4MRE: a benchmarking activity on the evaluation of Machine Reading systems through Question Answering and Reading Comprehension Test
- RepLab (new): a benchmarking activity on microblog data for online reputation management
- eHealth (new): a workshop on new evaluation issues in the health domain, related to the Louhi series of workshops on NLP in Health Informatics.

The Call for papers for the Conference was released December 2011; the expected deadline for the submission of papers is late April 2012.

Links:
CLEF 2012: http://www.clef2012.org/
CLEF: http://www.clef-campaign.org/
DIRECT: http://direct.dei.unipd.it/
PROMISE: http://www.promise-noe.eu/

Please contact:
Nicola Ferro
University of Padua, Italy
E-mail: ferro@dei.unipd.it

Photo: Ricardo Jota

INTERACT conference participants

Building Bridges - INTERACT 2011

by Joaquim Jorge

INESC-ID and The IFIP Technical Committee on Human-Computer Interaction (TC13) closed yet another successful instance of its bi-annual event last September.

INTERACT 2011 gathered close to 400 people in sunny Lisbon, from 5-9 September, culminating in a non-stop one week marathon with two main conferences, thirteen workshops, six tutorials and four keynote presentations during which the participants discussed their views on the design and use of current and future interactive systems. The event was organized by INESC-ID, part of the Portuguese ERCIM member PEG.

INTERACT's main theme this year was "Building Bridges". And many were built. From 2 to 4 September a singular conference called INTERACCIÓN brought together the Portuguese and Spanish communities, just before the main event.

Bridges were also built when participants from industry exchanged their view of the world with practitioners, usability professionals and scientists in engaging and vibrant dialogues. INTERACT also built bridges across the world - 400 participants from 43 different countries from around the world attended the main conference. The program featured a record number of 112 technical papers selected from 402 submissions (28% acceptance rate) and 60 short communications selected from 278 submissions (22% acceptance rate). These numbers represent increases of 12% in volume as reported to 2009, attested by the some 1800 pages of proceedings published by Springer LNCS. Indeed, such figures are witness to the prominent role and community engagement in the INTERACT event. No less than 800 different reviewers were called upon to select entries from almost 700 submissions to the overall technical program.

Outstanding keynote presentations by Mary Czerwinski (MS Research), António Camara (Ydreams), Don Norman (NNG) and Saul Greenberg (Calgary) served as capstone to this very successful event.

Photo: Ricardo Jota

Mary Czerwinski

INTERACT 2011 is also building bridges between the real and digital realms. The post-INTERACT initiative aims at continuing and extending the exchanges past the physical venue to continue the conference online, via video recordings of all technical sessions and debate forums that we hope will continue and enhance the dialogue between authors and the community.

The next conference will take place in Cape Town South Africa on September 2013. We hope to see you there!

More information:
http://interact2011.org

Strong W3C Presence at WWW2012

Lyon, France, 16-22 April 2012

At the 21st International World Wide Web Conference - WWW2012, W3C and its staff and members are organizing different activities to discuss and interact with the conference participants:

- 16-17 April: four half-day training courses: "CSS3 in Style"; "Get Up to Speed Quickly with Accessibility"; "Developing Mobile Web Applications"; and "Open Data in Practice".
- 18-19 April: two developers' camps related to "Web Security" and "HTML5 games".
- 19 April: keynote by Tim Berners-Lee.

Link:
http://www.w3.org/2012/04/w3c-track.html

ERCIM Dependable Embedded Systems Workshop at SAFECOMP 2012

Magdeburg, Germany, 25 September 2012

As in the previous years, the ERCIM Dependable Embedded Systems (DES) Working Group is organizing a workshop on Dependable Embedded Systems (Cyber-physical Systems) at SAFECOMP 2012.

Contributions to the workshop (papers of 6-10 pages) can be sent to the DES-WG chair Erwin Schoitsch (Erwin.schoitsch@ait.ac) until 19 February 2012. Topics are design, development, validation and certification of dependable embedded systems, robotics, autonomous systems, cyber-physical systems, including assessment, evaluation, standardization and education & training. The workshop is dedicated mainly to "work-in-progress", and may serve as dissemination event for European and national funded projects in the areas addressed. Attendance to the workshop is free of charge.

More information:
http://www-e.uni-magdeburg.de/safecomp/

EuroSys 2012

Bern, Switzerland 10-13 April, 2012

EuroSys 2012 is organized by EuroSys, the European Chapter of SIGOPS, sponsored by ACM SIGOPS. The EuroSys conference series brings together professionals from academia and industry. It has a strong focus on systems research and development: operating systems, data base systems, real-time systems and middleware for networked, distributed, parallel, or embedded computing systems. EuroSys has become a premier forum for discussing various issues of systems software research and development, including implications related to hardware and applications.

EuroSys 2012 will follow the pattern established by the previous EuroSys conferences, by seeking papers on all aspects of computer systems. EuroSys 2012 will also include a number of workshops to allow junior and senior members of the systems community to explore leading-edge topics and ideas before they are presented at a conference.

More information:
http://eurosys2012.unibe.ch

2012 International Zurich Seminar on Communications

Zurich, Switzerland, 29 February - 2 March 2012

The International Zurich Seminar on Communications (IZS) is a biennial conference with technical presentations in the broad area of telecommunications. The IZS is an opportunity to learn from, and to communicate with, leading experts in areas beyond one's own specialty. As in 2004, 2006, 2008 and 2010, roughly half of the presentations will be invited by external session organizers. No parallel sessions are anticipated, and all papers should be presented with a wide audience in mind. The IZS is organized by the IEEE Switzerland Chapter on Digital Communications in collaboration with ETH Zurich.

More information:
http://www.ifi.uzh.ch/icse2012/

Recruitment of Inria Researchers in 2012

At the interface of informatics and mathematics, 3,400 researchers located in Inria's eight research centers work, along with their academic and industrial partners from around the world, on the science and technologies of tomorrow. In 2012, there will be two different ways to join an Inria team:

- **20 tenured researcher and research director positions**, filled on the basis of competitive calls, intended for junior or senior profiles wanting to invest in medium- or long-term research.
- **8 research positions** consisting of three-year renewable contracts for innovative international profiles attracted by the excellence of the research produced at Inria: starting positions (after the thesis or a post-doctoral experience) or advanced positions (at least eight years' experience after the thesis). These positions are real career boosters giving you an opportunity to enrich your experience.

Recruitment campaign from 16 January to 17 February 2012
http://www.inria.fr/en

34th International Conference on Software Engineering

Zurich, Switzerland 2-9 June, 2012

The world's demand for software has become so huge that it can only be satisfied by building sustainable software. Conversely, a sustainable world needs more software than ever. Providing this software entails major challenges and opportunities for software engineering research and practice. ICSE, the International Conference on Software Engineering, is the premier software engineering conference, providing a forum for researchers, practitioners and educators to present and discuss the most recent innovations, trends, experiences and issues in the field of software engineering.

More information:
http://www.ifi.uzh.ch/icse2012/

W3DevCampus Hosts W3C Online Training Courses for Web Developers

Through its new W3DevCampus portal, W3C is pleased to announce a third edition of the most popular W3C online training course "W3C Introduction to Mobile Web and Application Best Practices". This 8-week course begins 30 January 2012. Developed by the W3C/MobiWebApp team, the course helps Web designers and content producers to quickly become mobile Web experts and create mobile-friendly sites. The course is led by trainers Frances de Waal and Phil Archer. During the course, participants will:

- learn about and use the recommended versions of HTML and CSS to use for mobile today
- understand the constraints of working on mobile and how to overcome them to deliver the best possible experience to the widest range of users
- practice client side and server side content adaptation techniques
- learn a bout and use the exciting new APIs available on modern mobile platforms.

Along with the full course description, read comments from past students and what they have achieved. An early bird rate of €165 is available until 9 January 2012.

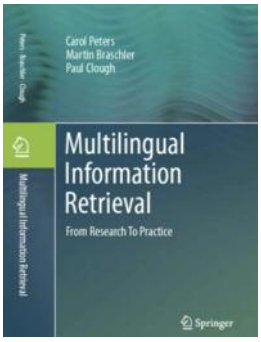
Also as part of the MobiWebApp project, a more advanced mobile Web course, titled "Writing Great Web Applications for Mobile", will be open for registration end of January 2012, with a start date of 5 March 2012. To get alerted, subscribe to the mailing list and/or follow @W3Training.

All courses follow a similar format: weekly modules consisting of lectures and links to further resources, followed by practical exercises such as quizzes and/or assignments. A discussion forum allows participants to discuss the course with each other and with the instructors.

Links:
W3DevCampus:
http://www.w3devcampus.com/
MWABP course:
http://www.w3devcampus.com/mobile-web-and-application-best-practices-training/
MobiWebApp EU project:
http://mobiwebapp.eu/
@W3Training:
http://twitter.com/#!/W3Training

Carol Peters, Martin Braschler, Paul Clough

Multilingual Information Retrieval: From Research to Practice



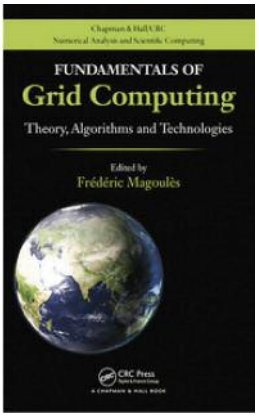
A comprehensive description of the technologies involved in designing and developing systems for Multilingual Information Retrieval (MLIR) is provided. Details on Cross-Language Information Retrieval (CLIR) are also covered that help readers to understand how to build systems that cross language boundaries. The book accompanies the reader step-by-step through the various stages involved in implementing, using and evaluating MLIR systems. It concludes with some examples of recent applications that utilise MLIR technologies. Some of the techniques described have recently started to appear in commercial search systems, while others have the potential to be part of future incarnations.

This book is intended for scholars, and practitioners with a basic understanding of classical text retrieval methods. It offers guidelines and information on all aspects that need to be taken into consideration when building MLIR systems, while avoiding too many 'hands-on details' that could rapidly become obsolete.

Springer, Database Management and Information Retrieval
Hardcover 217 pages.
ISBN 978-3-642-23007-3

Frédéric Magoulès

Fundamentals of Grid Computing Theory, Algorithms and Technologies



The integration and convergence of state-of-the-art technologies in the grid have enabled more flexible, automatic, and complex grid services to fulfill industrial and commercial needs, from the LHC at CERN to meteorological forecasting systems. Fundamentals of Grid Computing: Theory, Algorithms and Technologies discusses how the novel technologies of semantic web and workflow have been integrated into the grid and grid services. The book explains how distributed mutual exclusion algorithms offer solutions to transmission and control processes. It also addresses the replication problem in data grids with limited replica storage and the problem of data management in grids. After comparing utility, grid, autonomic, and cloud computing, the book presents efficient solutions for the reliable execution of applications in computational grid platforms. It then describes a fault tolerant distributed scheduling algorithm for large-scale distributed applications, along with broadcasting algorithms for institutional grids. The final chapter shows how load balancing is integrated into a real-world scientific application.

Chapman and Hall/CRC Numerical Analy and Scient Comp. Series; Hardback: 322 pages;
ISBN: 9781439803677, ISBN 10: 1439803676

Royal Decoration for Former CWI Director Jan Karel Lenstra

Jan Karel Lenstra was named Knight in the Order of the Netherlands Lion on 4 November 2011. This happened during the symposium on the occasion of his resignation as general director of CWI. The royal decoration is awarded to citizens with outstanding achievements in the field of arts, science, sports or music. The former CWI director was

Photo: Peter Lovie

Jan Karel Lenstra

praised for his pioneering and influential scientific achievements in mathematics, and for his efforts for the scientific community. He introduced successful new lines of research during his directorship, such as cryptology and life sciences, and was able to attract very talented researchers. In gratitude for his efforts, CWI management recently decided to appoint Lenstra as a CWI Fellow. Lenstra was general director of CWI from 2001 to 2011 and remains affiliated to CWI, where he will focus on research and administrative duties. Jan Karel Lenstra represented CWI on ERCIM's Board of Directors from 2004 to 2011 and was Vice-President of ERCIM from 2005 to 2009.

More information: http://www.cwi.nl/news/2011/royal-decoration-mathematician-and-former-cwi-director-jan-karel-lenstra

Inria Creates a Research and Innovation Centre in Chile

During the Chilean technology transfer week in October 2011, the Chilean economic development agency CORFO announced that Inria's project to create a research and innovation centre had been selected as part of a programme for the creation of "international centres of excellence for competitiveness" in Chile. The "Communication and Information Research and Innovation Center" (CIRIC) will allow scientists from different fields to work closely together and provide a support structure for technology transfer. It will make it possible to introduce a research and development culture into universities in partnership with business, in order to support Chile's competitiveness in computational science and technologies, particularly through the creation of spin-offs and technology transfer to large companies and SMEs.

The Centre will be based in Santiago, with a branch in Valparaíso. It will initially be built in partnership with nine Chilean universities and focus on three lines of research and development: Internet and telecommunications networks, Management of natural resources, and Hybrid energy sources.

The purpose of the CIRIC is to host and catalyse all of Inria's collaborations in Chile and provide the institute with an initial base in Latin America. It will be open to international partnerships in particular in Latin America and Europe, and especially with those European institutions setting up a base in Chile as part of the programme, namely the ERCIM member Fraunhofer-Gesellschaft and the Australian CSIRO.

More information:
http://www.inria.fr/en/news/news-from-inria/inria-centre-in-chile

Award of the Academy of Sciences of the Czech Republic

A research group from the Institute of Information Theory and Automation (UTIA-CRCIM) won the prestigious Czech Republic scientific award - the Award of the Academy of

Award ceremony

Sciences of the Czech Republic for outstanding results of major scientific importace. The team lead by Prof. Michal Haindl consisting of Jiří Filip, Jiří Grim, Vojtěch Havlíček, and Martim Hatka was awarded for the scientific outcome of "Mathematical modeling of visual properties of surface materials". A short article about this research was published in ERCIM News no. 81 (http://ercim-news.ercim.eu/en81/special/realistic-material-appearance-modelling). The award was handed to the researchers by the president of the Academy of Sciences of the Czech Republic Jiří Drahoš in a ceremony on 22 September 2011 in Prague.

ERCIM
European Research Consortium for Informatics and Mathematics

ERCIM – the European Research Consortium for Informatics and Mathematics is an organisation dedicated to the advancement of European research and development, in information technology and applied mathematics. Its member institutions aim to foster collaborative work within the European research community and to increase co-operation with European industry.

ERCIM is the European Host of the World Wide Web Consortium.

AARIT
Austrian Association for Research in IT
c/o Österreichische Computer Gesellschaft
Wollzeile 1-3, A-1010 Wien, Austria
http://www.aarit.at/

CNR
Consiglio Nazionale delle Ricerche, ISTI-CNR
Area della Ricerca CNR di Pisa,
Via G. Moruzzi 1, 56124 Pisa, Italy
http://www.isti.cnr.it/

Crcim
Czech Research Consortium
for Informatics and Mathematics
FI MU, Botanicka 68a, CZ-602 00 Brno, Czech Republic
http://www.utia.cas.cz/CRCIM/home.html

CWI
Centrum Wiskunde & Informatica
Science Park 123,
NL-1098 XG Amsterdam, The Netherlands
http://www.cwi.nl/

Fonds National de la Recherche Luxembourg
Fonds National de la Recherche
6, rue Antoine de Saint-Exupéry, B.P. 1777
L-1017 Luxembourg-Kirchberg
http://www.fnr.lu/

FWO
Egmontstraat 5
B-1000 Brussels, Belgium
http://www.fwo.be/

FNRS
rue d'Egmont 5
B-1000 Brussels, Belgium
http://www.fnrs.be/

FORTH
Foundation for Research and Technology – Hellas
Institute of Computer Science
P.O. Box 1385, GR-71110 Heraklion, Crete, Greece
http://www.ics.forth.gr/

Fraunhofer IUK-TECHNOLOGIE
Fraunhofer ICT Group
Friedrichstr. 60
10117 Berlin, Germany
http://www.iuk.fraunhofer.de/

Inria
Institut National de Recherche en Informatique
et en Automatique
B.P. 105, F-78153 Le Chesnay, France
http://www.inria.fr/

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology, Mathematics and
Electrical Engineering, N 7491 Trondheim, Norway
http://www.ntnu.no/

PEG
Portuguese ERCIM Grouping
c/o INESC Porto, Campus da FEUP,
Rua Dr. Roberto Frias, nº 378,
4200-465 Porto, Portugal

Universitas Varsoviensis PLERCIM Universitas Wratislaviensis
Polish Research Consortium for Informatics and Mathematics
Wydział Matematyki, Informatyki i Mechaniki,
Uniwersytetu Warszawskiego, ul. Banacha 2, 02-097 Warszawa, Poland
http://www.plercim.pl/

Science & Technology Facilities Council
Science and Technology Facilities Council,
Rutherford Appleton Laboratory
Harwell Science and Innovation Campus
Chilton, Didcot, Oxfordshire OX11 0QX, United Kingdom
http://www.scitech.ac.uk/

SpaRCIM
Spanish Research Consortium for Informatics and Mathematics,
D3301, Facultad de Informática, Universidad Politécnica de Madrid,
Campus de Montegancedo s/n,
28660 Boadilla del Monte, Madrid, Spain,
http://www.sparcim.es/

Swedish Institute of Computer Science SICS
Swedish Institute of Computer Science
Box 1263,
SE-164 29 Kista, Sweden
http://www.sics.se/

SARIT
Swiss Association for Research in Information Technology
c/o Professor Daniel Thalmann, EPFL-VRlab,
CH-1015 Lausanne, Switzerland
http://www.sarit.ch/

MTA SZTAKI
Magyar Tudományos Akadémia
Számítástechnikai és Automatizálási Kutató Intézet
P.O. Box 63, H-1518 Budapest, Hungary
http://www.sztaki.hu/

University of Cyprus
P.O. Box 20537
1678 Nicosia, Cyprus
http://www.cs.ucy.ac.cy/

VTT
Technical Research Centre of Finland
PO Box 1000
FIN-02044 VTT, Finland
http://www.vtt.fi/

Order Form

If you wish to subscribe to ERCIM News ***free of charge*** *or if you know of a colleague who would like to receive regular copies of ERCIM News, please fill in this form and we will add you/them to the mailing list.*

Send, fax or email this form to:
ERCIM NEWS
2004 route des Lucioles
BP 93
F-06902 Sophia Antipolis Cedex
Fax: +33 4 9238 5011
E-mail: contact@ercim.eu

Data from this form will be held on a computer database. By giving your email address, you allow ERCIM to send you email

I wish to subscribe to the

☐ ***printed edition*** ☐ ***online edition (email required)***

Name:

Organisation/Company:

Address:

Postal Code:

City:

Country

E-mail:

You can also subscribe to ERCIM News and order back copies by filling out the form at the ERCIM Web site at http://ercim-news.ercim.eu/