IN PARTNERSHIP WITH:
CNRS

Ecole nationale supérieure d'électronique, informatique et radiocommunications de Bordeaux

Université de Bordeaux

Activity Report 2011

Project-Team PHOENIX

Programming Language Technology For Communication Services

IN COLLABORATION WITH: Laboratoire Bordelais de Recherche en Informatique (LaBRI)

RESEARCH CENTER
Bordeaux - Sud-Ouest

THEME
Distributed Systems and Services

Table of contents

Project-Team PHOENIX

Keywords: Programming Languages, Software Engineering, Domain-Specific Languages

1. Members

Research Scientist

Emilie Balland [Junior Researcher, Inria]

Faculty Members

Charles Consel [Team Leader, Professor, ENSEIRB, HdR]
Bernard N'Kaoua [Professor, University Bordeaux Segalen (Associate Member)]
Hélène Sauzéon [Associate Professor, University Bordeaux Segalen (Associate Member)]

External Collaborator

Julia Lawall [Associate Professor, University of Copenhagen (DIKU)]

Technical Staff

Ghislain Deffrasnes [Associate Engineer, from October 1, 2009 to September 11, 2011]
Benjamin Bertran [Associate Engineer, from December 17, 2007 to December 18, 2011]
Amélie Marzin [Associate Engineer, from October 3, 2011]

PhD Students

Julien Mercadal [Ministerial scholarship, from October 1, 2006 to January 31, 2011, University of Bordeaux and Research Assistant from April 1, 2011 to September 30, 2011, Inria]
Damien Cassou [Ministerial scholarship, from October 1, 2007 to September 30, 2010, University of Bordeaux and Research Assistant, from October 1, 2010 to March 31, 2011, Inria]
Henner Jakob [Inria scholarship, from May 1, 2008 to June 30, 2011]
Julien Bruneau [Thales scholarship, from October 1, 2008 to September 30, 2011 and Research Assistant from October 1, 2011, Inria]
Hongyu Guan [Region scholarship, from February 9, 2009]
Pengfei Liu [Inria scholarship, from October 1, 2009]
Stéphanie Gatti [Thales scholarship, from February 1, 2010]
Quentin Enard [Thales scholarship, from February 1, 2010]
Luc Vercellin [Inria scholarship, from October 1, 2011]

Post-Doctoral Fellows

Christine Louberry [Post-Doctoral Fellow, from September 1, 2011]
Young-Joo Moon [Post-Doctoral Fellow, from November 2, 2011]

Administrative Assistant

Sylvie Embolla [Group Assistant, from September 4, 2006]

2. Overall Objectives

2.1. Overall Objectives

A host of networked devices are populating smart spaces that become prevalent in an increasing number of areas, including supply chain management (e.g., parcel tracking), monitoring (e.g., building surveillance and patient monitoring) and home and building automation (e.g., control of energy consumption). This situation raises a number of challenges (1) safety and security because of the interweaving of these smart spaces in our daily life, (2) productivity because of a high demand of applications matching the wide range of user needs, and (3) abstraction because of the heterogeneity of the devices.

To address these challenges, we develop a software engineering approach that is dedicated to services orchestrating networked devices:

- the specification of robust orchestrating services based on innovative Domain-Specific Languages (DSLs),
- the study of the communication layers underlying these services to improve flexibility and performance,
- the application to concrete areas such as pervasive computing or avionics to validate our approach.

2.2. Highlights

- Launching of a research activity on digital cognitive assistance (two associate members in cognitive science, starting of a PhD thesis and an ADT, national and international collaborations)
- Technology transfer action for DiaSuiteBox (CSATT support, demonstrations and showcases, starting of partnerships)
- Three PhDs defended in 2011 (Damien Cassou, Henner Jakob and Julien Mercadal)
- Organization of the International IFIP Working Conference on Domain-Specific Languages (DSL 2011) and the meeting of the Working Group IFIP 2.11 in Bordeaux

3. Scientific Foundations

3.1. Introduction

Our proposed project builds upon results previously obtained by the Compose research group whose aim was to study new approaches to developing adaptable software components in the domain of systems and networking. In this section, we review the accomplishments of Compose, only considering the ones achieved by the current project members, to demonstrate our expertise in the key areas underlying our project, namely:

- Programming language technology: language design and implementation, domain-specific languages, program analysis and program transformation.
- Operating Systems and Networking: design, implementation and optimization.
- Software engineering: software architecture, methodologies, techniques and tools.

By combining expertise in these areas, the research work of the Compose group contributed to demonstrating the usefulness of adaptation methodologies, such as domain-specific languages, and the effectiveness of adaptation tools, such as program specializers. Our work aimed to show how adaptation methodologies and tools could be integrated into the development process of real-size software components. This contribution relied on advances in methodologies to develop adaptable programs, and techniques and tools to adapt these programs to specific usage contexts.

3.2. Adaptation Methodologies

Although industry has long recognized the need to develop adaptable programs, methodologies to develop them are still at the research stage. We have presented preliminary results in this area with a detailed study of the applicability of program specialization to various software architectures [31]. Our latest contributions in this area span from a revolutionary approach based on the definition of programming languages, dedicated to a specific problem family, to a direct exploitation of specialization opportunities generated by a conventional programming methodology.

3.2.1. Domain-Specific languages

DSLs represent a promising approach to modeling a problem family. Yet, this approach currently suffers from the lack of methodology to design and implement DSLs. To address this basic need, we have introduced the Sprint methodology for DSL development [23]. This methodology bridges the gap between semantics-based approaches to developing general-purpose languages and software engineering. Sprint is a complete software development process starting from the identification of the need for a DSL to its efficient implementation. It uses the denotational framework to formalize the basic components of a DSL. The semantic definition is structured so as to stage design decisions and to smoothly integrate implementation concerns.

3.2.2. Declaring adaptation

A less drastic strategy to developing efficient adaptable programs consists of making specific issues of adaptation explicit via a declarative approach. To do so, we enrich Java classes with declarations, named *adaptation classes*, aimed to express adaptive behaviors [20]. As such, this approach allows the programmer to separate the concerns between the basic features of the application and its adaptation aspects. A dedicated compiler automatically generates Java code that implements the adaptive features.

3.2.3. Declaring specialization

When developing components, programmers often hesitate to make them highly generic and configurable. Indeed, genericity and configurability systematically introduce overheads in the resulting component. However, the causes of these overheads are usually well-known by the programmers and their removal could often be automated, if only they could be declared to guide an optimizing tool. The Compose group has worked towards solving this problem.

We introduced a declaration language which enables a component developer to express the configurability of a component. The declarations consist of a collection of specialization scenarios that precisely identify what program constructs are of interest for specialization. The scenarios of a component do not clutter the component code; they are defined aside in a *specialization module* [26], [27], [25], [28].

This work was done in the context of C and declarations were intended to drive our C specializer.

3.2.4. Specializing design patterns

A natural approach to systematically applying program specialization is to exploit opportunities offered by a programming methodology. We have studied a development methodology for object-oriented languages, called design patterns. Design patterns encapsulate knowledge about the design and implementation of highly adaptable software. However, adaptability is obtained at the expense of overheads introduced in the finished program. These overheads can be identified for each design pattern. Our work consisted in using knowledge derived from design patterns to eliminate these overheads in a systematic way. To do so, we analyzed the specialization opportunities provided by specific uses of design patterns, and determined how to eliminate these overheads using program specialization. These opportunities were documented in declarations, called specialization patterns, and were associated with specific design patterns [39]. The specialization of a program composed of design patterns was then driven by the corresponding declarations. This work was presented in the context of Java and uses our Java specializer [38].

3.2.5. Specializing software architectures

The sources of inefficiency in software architectures can be identified in the data and control integration of components, because flexibility is present not only at the design level but also in the implementation. We proposed the use of program specialization in software engineering as a systematic way to improve performance and, in some cases, to reduce program size. We studied several representative, flexible mechanisms found in software architectures: selective broadcast, pattern matching, interpreters, layers and generic libraries. We showed how program specialization can be applied systematically to optimize these mechanisms [30], [31].

3.3. Adaptation in Systems Software

3.3.1. DSLs in Operating Systems

Integrating our adaptation methodologies and tools into the development process of real-size software systems was achieved by proposing a new development process. Specifically, we proposed a new approach to designing and structuring operating systems (OSes) [34]. This approach was based on DSLs and enables rapid development of robust OSes. Such an approach is critically needed in application domains, like appliances, where new products appear at a rapid pace and needs are unpredictable.

3.3.2. Devil - a DSL for device drivers

Our approach to developing systems software applied to the domain of device drivers. Indeed, peripheral devices come out at a frantic pace, and the development of drivers is very intricate and error prone. The Compose group developed a DSL, named Devil (DEvice Interface Language), to solve these problems; it was dedicated to the basic communication with a device. Devil allowed the programmer to easily map device documentation into a formal device description that can be verified and compiled into executable code.

From a software engineering viewpoint, Devil captures domain expertise and systematizes re-use because it offers suitable built-in abstractions [36]. A Devil description formally specifies the access mechanisms, the type and layout of data, as well as behavioral properties involved in operating the device. Once compiled, a Devil description implements an interface to an idealized device and abstracts the hardware intricacies.

From an operating systems viewpoint, Devil can be seen as an *interface definition language* for hardware functionalities. To validate the approach, Devil was put to practice [35]: its expressiveness was demonstrated by the wide variety of devices that have been specified in Devil. No loss in performance was found for the compiled Devil description compared to an equivalent C code.

From a dependable system viewpoint, Devil improves safety by enabling descriptions to be statically checked for consistency and generating stubs including additional run-time checks [37]. Mutation analysis was used to evaluate the improvement in driver robustness offered by Devil. Based on our experiments, Devil specifications were found up to 6 times less prone to errors than writing C code.

Devil was the continuation of a study of graphic display adaptors for a X11 server. We developed a DSL, called GAL (Graphics Adaptor Language), aimed to specify device drivers in this context [42]. Although covering a very restricted domain, this language was a very successful proof of concept.

3.4. Adaptation Tools and Techniques

To further the applicability of our approach, we have strengthened and extended adaptation tools and techniques. We have produced a detailed description of the key program analysis for imperative specialization, namely binding-time analysis [22]. This analysis is at the heart of our program specializer for C, named Tempo [22]. We have examined the importance of the accuracy of these analyses to successfully specialize existing programs. This study was conducted in the context of systems software [32].

Tempo is the only specializer which enables programs to be specialized both at compile time and run time. Yet, specialization is always performed in one stage. As a consequence, this process cannot be factorized even if specialization values become available at multiple stages. We present a realistic and flexible approach to achieving efficient incremental run-time specialization [29]. Rather than developing new techniques, our strategy for incremental run-time specialization reuses existing technology by iterating a specialization process. Our approach has been implemented in Tempo.

While program specialization encodes the result of early computations into a new program, *data specialization* encodes the result of early computations into data structures. Although aiming at the same goal, namely processing early computations, these two forms of specialization have always been studied separately. The Compose group has proposed an extension of Tempo to perform both program and data specialization [21]. We showed how these two strategies can be integrated in a single specializer. Most notably, having both strategies enabled us to assess their benefits, limitations and their combination on a variety of programs.

Interpreters and run-time compilers are increasingly used to cope with heterogeneous architectures, evolving programming languages, and dynamically loaded code. Although solving the same problem, these two strategies are very different. Interpreters are simple to implement but yield poor performance. Run-time compilation yields better performance, but is costly to implement. One approach to reconciling these two strategies is to develop interpreters for simplicity but to use specialization to achieve efficiency. Additionally, a specializer like Tempo can remove the interpretation overhead at compile time as well as at run time. We have conducted experiments to assess the benefits of applying specialization to interpreters [41]. These experiments have involved Bytecode and structured-language interpreters. Our experimental data showed that specialization of structured-language interpreters can yield performance comparable to that of the compiled code of an optimizing compiler.

Besides targeting C, we developed the first program specializer for an object-oriented language. This specializer, named JSpec, processes Java programs [38]. JSpec is constructed from existing tools. Java programs are translated into C using our Java compiler, named Harissa. Then, the resulting C programs are specialized using Tempo. The specialized C program is executed in the Harissa environment. JSpec has been used for various applications and has shown to produce significant speedups [40].

4. Application Domains

4.1. Introduction

After having explored DSLs in isolated domains in the past, we now generalize this experience to attack a larger domain, namely, communication services. Generalizing our work on telephony, we investigated the coordination of *networked entities*, whether or not operated by users. The three main application domains are pervasive computing, avionics and assisted living.

4.2. Pervasive Computing

Pervasive computing systems are being deployed in a rapidly increasing number of areas, including building automation and supply chain management. Regardless of their target area, pervasive computing systems have a typical architectural pattern. They aggregate data from a variety of distributed sources, whether sensing devices or software components, analyze a context to make decisions, and carry out decisions by invoking a range of actuators. Because pervasive computing systems are standing at the crossroads of several domains (*e.g.,* distributed systems, multimedia, and embedded systems), they raise a number of challenges in software development:

- *Heterogeneity.* Pervasive computing systems are made of off-the-shelf entities, that is, hardware and software building blocks. These entities run on specific platforms, feature various interaction models, and provide non-standard interfaces. This heterogeneity tends to percolate in the application code, preventing its portability and reusability, and cluttering it with low-level details.
- *Lack of structuring.* Pervasive computing systems coordinate numerous, interrelated components. A lack of global structuring makes the development and evolution of such systems error-prone: component interactions may be invalid or missing.
- *Combination of technologies.* Pervasive computing systems involve a variety of technological issues, including device intricacies, complex APIs of distributed systems technologies and middleware-specific features. Coping with this range of issues results in code bloated with special cases to glue technologies together.
- *Dynamicity.* In a pervasive computing system, devices may either become available as they get deployed, or unavailable due to malfunction or network failure. Dealing with these issues explicitly in the implementation can quickly make the code cumbersome.
- *Testing.* Pervasive computing systems are complicated to test. Doing so requires equipments to be acquired, tested, configured and deployed. Furthermore, some scenarios cannot be tested because of the nature of the situations involved (*e.g.,* fire and smoke). As a result, the programmer must resort to writing specific code to achieve ad hoc testing.

4.3. Avionics

In avionics, an aircraft can be seen as an environment full of sensors (*e.g.,* accelerometers, gyroscopes, and GPS sensors) and actuators (*e.g.,* ailerons and elevator trim). For example, a flight guidance system controls the aircraft using data produced by sensors. In a critical platform such as an aircraft, software systems have to be certified. Moreover the safety-critical nature of the avionics domain takes the form of stringent non-functional requirements, resulting in a number of challenges in software development:

- *Traceability.* Traceability is the ability to trace all the requirements throughout the development process. In the avionics certification processes, traceability is mandatory for both functional and non-functional requirements.
- *Coherence.* Functional and non-functional aspects of an application are inherently coupled. For example, dependability mechanisms can potentially deteriorate the overall performance of the application. The coherence of the requirements is particularly critical when the software evolves: even minor modifications to one aspect may tremendously impact the others, leading to unpredicted failures.
- *Separation of concerns.* Avionics platforms involve the collaboration of several experts (from low-level system to software, safety, QoS), making requirements traceability significantly more challenging. Providing development methodologies that allow a clear separation of concerns can tremendously improve traceability.

Our approach consists of enriching a design language with non-functional decalarations. Such declarations allow the safety expert to specify at design time how errors are handled, guiding and facilitating the implementation of error handling code. The design is also enriched with Quality of Service (QoS) declarations such as time constraints. For each of these non-functional declarations, specific development support can be generated. We have validated this approach by developing flight guidance applications for avionics and drone systems.

4.4. Assisted Living

Cognitive impairments (memory, attention, time and space orientation, *etc*) affect a large part of the population, including elderly, patients with brain injuries (traumatic brain injury, stroke, *etc*), and people suffering from cognitive disabilities, such as Down syndrome.

The emerging industry of digital assistive technologies provide hardware devices dedicated to specific tasks, such as a telephone set with a keyboard picturing relatives (http://www.doro.fr), or a device for audio and video communication over the web (http://www.technosens.fr). These assistive technologies apply a traditional approach to personal assistance by providing an equipment dedicated to a single task (or a limited set of tasks), without leveraging surrounding devices. This traditional approach has fundamental limitations that must be overcome to significantly improve assistive technologies:

- they are *not adaptable to one's needs*. They are generally dedicated to a task and have very limited functionalities: no networking, limited computing capabilities, a limited screen and rudimentary interaction modalities. This lack of functionality may cause a proliferation of devices, complicating the end-user life. Moreover, they are rarely designed to adapt to the cognitive changes of the user. When the requirements evolve, the person must acquire a new device.
- they are often *proprietary*, limiting innovation. As a result, they cannot cope with the evolution of users' needs.
- they have limited or *no interoperability*. As a result, they cannot rely on other devices and software services to offer richer applications.

To break this model, we propose to offer an assistive solution that is open-ended in terms of applications and entities. (1) An on-line catalog of available applications enables every user and caregiver to define personalized assistance in the form of an evolving and adapted set of applications; this catalog provides a community of developers with a mechanism to publish applications for specific daily-activity needs. (2) New types of entities can be added to a platform description to enhance its functionalities and extend the scope of future applications.

5. Software

5.1. DiaSuite: a Development Environment for Sense/Compute/Control Applications

Participants: Charles Consel [correspondent], Benjamin Bertran, Ghislain Deffrasnes, Amélie Marzin, Damien Cassou, Julien Bruneau, Emilie Balland.

Despite much progress, developing a pervasive computing application remains a challenge because of a lack of conceptual frameworks and supporting tools. This challenge involves coping with heterogeneous devices, overcoming the intricacies of distributed systems technologies, working out an architecture for the application, encoding it in a program, writing specific code to test the application, and finally deploying it.

DIASUITE is a suite of tools covering the development life-cycle of a pervasive computing application:

- *Defining an application area.* First, an expert defines a catalog of entities, whether hardware or software, that are specific to a target area. These entities serve as building blocks to develop applications in this area. They are gathered in a taxonomy definition, written in the taxonomy layer of the DIASPEC language.
- *Designing an application.* Given a taxonomy, the architect can design and structure applications. To do so, the DIASPEC language provides an application design layer [33]. This layer is dedicated to an architectural pattern commonly used in the pervasive computing domain [24]. Describing the architecture application allows to further model a pervasive computing system, making explicit its functional decomposition.
- *Implementing an application.* We leverage the taxonomy definition and the architecture description to provide dedicated support to both the entity and the application developers. This support takes the form of a Java programming framework, generated by the DIAGEN compiler. The generated programming framework precisely guides the developer with respect to the taxonomy definition and the architecture description. It consists of high-level operations to discover entities and interact with both entities and application components. In doing so, it abstracts away from the underlying distributed technologies, providing further separation of concerns.
- *Testing an application.* DIAGEN generates a simulation support to test pervasive computing applications before their actual deployment. An application is simulated in the DIASIM tool, without requiring any code modification. DIASIM provides an editor to define simulation scenarios and a 2D-renderer to monitor the simulated application. Furthermore, simulated and actual entities can be mixed. This hybrid simulation enables an application to migrate incrementally to an actual environment.
- *Deploying a system.* Finally, the system administrator deploys the pervasive computing system. To this end, a distributed systems technology is selected. We have developed a back-end that currently targets the following technologies: Web Services, RMI, SIP and OSGI. This targeting is transparent for the application code. The variety of these target technologies demonstrates that our development approach separates concerns into well-defined layers.

This development cycle is summarized in the Figure 1.

See also the web page http://diasuite.inria.fr.

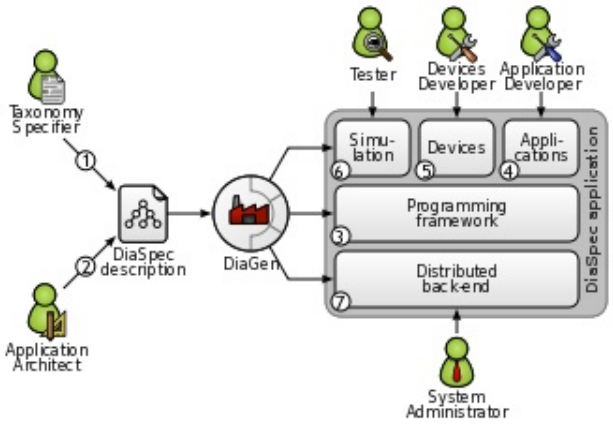

Figure 1. DIASUITE *Development Cycle*

5.1.1. DiaSpec: a Domain-Specific Language for Networked Entities

The core of the DIASUITE development environment is the domain specific language called DIASPEC and its compiler DIAGEN:

- DIASPEC is composed of two layers:
  - The *Taxonomy Layer* allows the declaration of entities that are relevant to the target application area. An entity consists of sensing capabilities, producing data, and actuating capabilities, providing actions. Accordingly, an entity description declares a data source for each one of its sensing capabilities. As well, an actuating capability corresponds to a set of method declarations. An entity declaration also includes attributes, characterizing properties of entity instances. Entity declarations are organized hierarchically allowing entity classes to inherit attributes, sources and actions. A taxonomy allows separation of concerns in that the expert can focus on the concerns of cataloging area-specific entities. The entity developer is concerned about mapping a taxonomical description into an actual entity, and the application developer concentrates on the application logic.
  - The *Architecture Layer* is based on an architectural pattern commonly used in the pervasive computing domain [24]. It consists of context components fueled by sensing entities. These components process gathered data to make them amenable to the application needs. Context data are then passed to controller components that trigger actions on entities. Using an architecture description enables the key components of an application to be identified, allowing their implementation to evolve with the requirements (*e.g.,* varying light management implementations in a controller component to optimize energy consumption).
- DIAGEN is the DIASPEC compiler that performs both static and runtime verifications over DIASPEC declarations and produces a dedicated programming framework that guides and eases the implementation of components. The generated framework is independent of the underlying distributed technology. As of today, DIAGEN supports multiple targets: Local, RMI, SIP, Web Services and OSGI.

5.1.2. DiaSim: a Parametrized Simulator for Pervasive Computing Applications

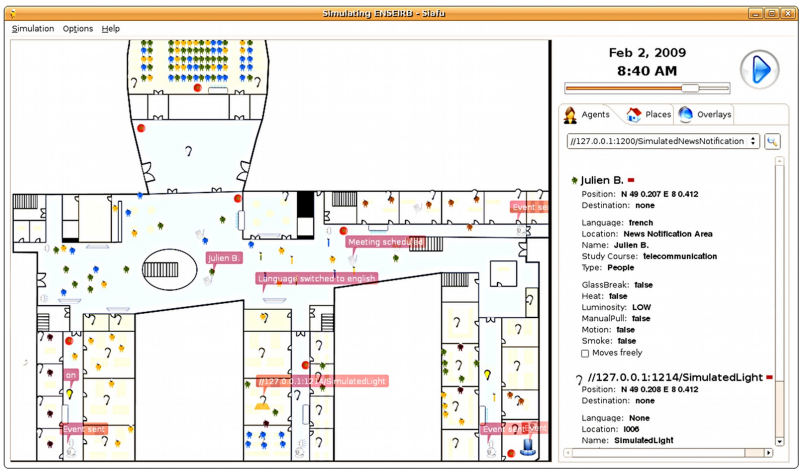

Figure 2. A screenshot of the DIASIM *simulator*

Pervasive computing applications involve both software and integration concerns. This situation is problematic for testing pervasive computing applications because it requires acquiring, testing and interfacing a variety of software and hardware entities. This process can rapidly become costly and time-consuming when the target environment involves many entities.

To ease the testing of pervasive applications, we are developing a simulator for pervasive computing applications: DIASIM. To cope with widely heterogeneous entities, DIASIM is parameterized with respect to a DIASPEC specification describing a target pervasive computing environment. This description is used to generate with DIAGEN both a programming framework to develop the simulation logic and an emulation layer to execute applications. Furthermore, a simulation renderer is coupled to DIASIM to allow a simulated pervasive system to be visually monitored and debugged. The simulation renderer is illustrated in Figure 2.

5.2. DiaSuiteBox: an Open Service Platform

Participants: Benjamin Bertran [correspondent], Julien Bruneau, Charles Consel, Emilie Balland.

The DiaSuiteBox platform runs an open-ended set of applications leveraging a range of appliances and web services. Our solution consists of a dedicated development environment, a certifying application store, and a lightweight runtime platform. This solution is based on the DIASUITE project.

The DiaSuiteBox platform can be embedded in a small plug-computer. This box can be easily deployed, runs silently, and has a reduced energy consumption. Thanks to the application store and the developer community, the platform is fed by a full offer of new innovative applications. During the submission process, an application is automatically analyzed and checked in order to be certified. The user is ensured of the behavior of its applications are innocuous and correct beside the provided information. This box relies on several technology standards like UPnP, Bluetooth, USB, etc. As shown in Figure 3, this platform can be easily extended by plugging appliances directly on the box or by connecting devices on the local network.

See also the web page http://diabox.inria.fr.

5.3. Pantagruel: a Visual Domain-Specific Language for Ubiquitous Computing

Participants: Ghislain Deffrasnes [correspondent], Julien Mercadal, Charles Consel.

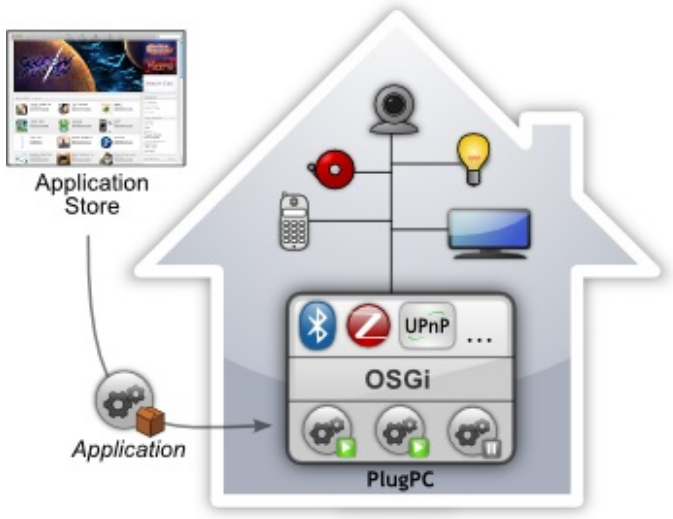

Figure 3. DiaSuiteBox platform architecture

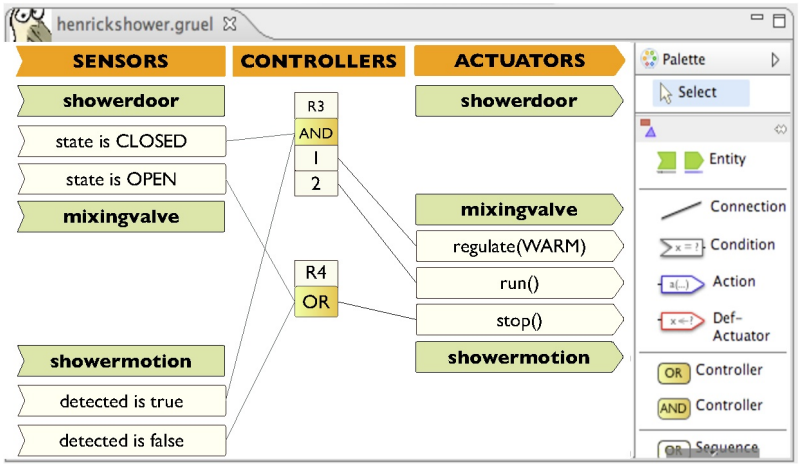

Figure 4. A screenshot of the Pantagruel graphical editor

Pantagruel aims at easing the description of an orchestration logic between networked entities of a pervasive environment. First, the developer defines a taxonomy of entities that compose the environment, This step provides an abstraction of the entities capabilities and functionalities. Second, the developer defines the orchestration logic in terms of rules. To facilitate its programming, we provide a visual domain-specific language based on the sensor-controller-actuator paradigm. An example of a visual orchestration is given in Figure 4 where a shower automatically runs at the right temperature when someone enters the bathroom and closes the door.

Pantagruel brings a high-level layer intended to complement existing tools in the activity of safe orchestration logic description, allowing novice-programmers to prototype pervasive applications. The Pantragruel compiler generates code compliant with the DIASUITE toolset. Pantagruel is being completed by tools aimed at verifying safety properties like termination and reachability.

See also the web page http://phoenix.inria.fr/software/pantagruel.

6. New Results

6.1. Leveraging Software Architectures to Guide and Verify the Development of Sense/Compute/Control Applications

A software architecture describes the structure of a computing system by specifying software components and their interactions. Mapping a software architecture to an implementation is a well known challenge. A key element of this mapping is the architecture's description of the data and control-flow interactions between components. The characterization of these interactions can be rather abstract or very concrete, providing more or less implementation guidance, programming support, and static verification.

In this work, we have introduced a notion of *behavioral contract* that expresses the set of allowed interactions between components, describing both data and control-flow constraints [15]. This declaration is part of the architecture description, allows generation of extensive programming support, and enables various verifications. We have instantiated our approach in an architecture description language for the domain of Sense/Compute/Control (SCC) applications, and described associated compilation and verification strategies.

The main contributions of this work are the following:

- We have introduced a language for behavioral contracts dedicated to SCC applications.
- We have shown that behavioral contracts can effectively guide the implementation of SCC applications by enabling the generation of highly customized programming frameworks using a dedicated compiler. This approach ensures the conformance between the architecture and the implementation, while facilitating software evolution.
- We have shown that such descriptions are precise enough to verify safety properties such as information flow reachability or behavioral invariants.
- Based on an implementation of behavioral contracts in a design language targeting SCC applications, we haved assessed the benefit of behavioral contracts at a conceptual level and in terms of metrics on the resulting code.

6.2. A Step-wise Approach for Integrating QoS throughout Software Development

Non-functional requirements are used to express the *quality* to be expected from a system. For real-time systems such as avionics, it is critical to guarantee this quality, in particular time-related performance properties. In this domain, deterministic QoS is generally ensured at the execution platform level (*e.g.,* operating systems, distributed systems technologies, hardware specificities), independently of a particular application. When addressing the QoS requirements of a given application, these platform-specific guarantees are not sufficient.

In this work, we have proposed a step-wise QoS approach integrated through all development phases and development artifacts [17]. This approach is dedicated to control-loop systems. Control-loop systems are systems that sense the external environment, compute data, and eventually control the environment accordingly. This kind of systems can be found in a range of domains, including avionics, robotics, and pervasive computing. For example, in the avionics domain, a flight management application is a control-loop system that (1) senses the environment for location and other navigation information, (2) computes the trajectory and (3) modifies the wings configuration accordingly.

The main contributions of this work are the following:

- We have developed a step-wise approach that systematically processes QoS requirements throughout software development. This integrated approach is dedicated to control-loop systems, allowing to rely on a particular architectural pattern and thus enhancing the design and programming support level for non-functional aspects. For now, we focus on time-related performance but the approach could be generalized to other non-functional properties (*e.g.,* CPU or memory consumption).
- Our approach has been integrated into DIASUITE , a tool-based development methodology dedicated to control-loop systems. DIASUITE is based on a dedicated design language that we have enriched with time-related performance properties. This non-functional extension has been used to offer verification and programming support at each development stage.
- Our approach has been applied to the development of avionics applications such as a flight management system and a collision avoidance system. These experiments have demonstrated that our step-wise approach can effectively guide the avionics certification process.

6.3. Architecturing Conflict Handling of Pervasive Computing Resources

The rapid development of new devices (further resources) and development tools being opened to third-parties have paved the way to an increasing number of applications being deployed in pervasive computing environments. These applications anarchically access resources. In this situation, it is very common for a resource to be accessed by multiple applications, potentially leading to conflicts. For example, in a building management system, a security application that grants access inside the building, can conflict with an application dealing with emergency situations like fires, preventing the building to be evacuated.

Managing conflicts consists of three main parts, detection, resolution and prevention. These parts crosscut the development cycle of applications and pervasive computing systems. In this work, we have proposed a conflict management process that cleanly separates conflict management tasks by providing a design method and supporting tools [18]. This facilitates the work of developers, architects and administrators, who can follow clear guidelines to manage conflicts.

The main contributions of this work are the following:

- We have identified requirements at different stages during the development cycle that are necessary to detect, resolve, and prevent conflicts. We have assigned duties and responsibilities to existing roles, that are carried out during the *conflict management process* without interfering with the standard application development.
- We have extended a domain-specific design language to declare conflict resolution at an architectural level. During the conflict management process conditions are specified and prioritized. Afterwards conflicting applications (inter application) or modules (intra application) are linked to these conditions.
- The declared information is used to generate code dedicated to conflict handling. On the one hand, a compiler generates a dedicated framework that guides the implementation of the conflict handling logic at application and system level. On the other hand, it generates code that orchestrates resource accesses and prevents conflicts.

7. Contracts and Grants with Industry

7.1. Designing and developing simulation capabilities for network-centric systems – Industrial Fellowship (CIFRE / Thales)

Participants: Charles Consel, Julien Bruneau.

The goal of this project is to provide simulation capabilities for testing network-centric systems. To achieve this goal, a formal description of the component behavior of such systems must be defined. Hybrid testing (combining virtual and real) of components, data and scenarios, as well as observability tools for component-tools will be studied in this project.

Models, DSLs and protocols for modeling a network-centric system will be designed and developed during this project. A dedicated framework for the simulation must also be provided. Finally, the simulation of a system must allow to qualify the functional logic of this system.

7.2. Integrating non-functional properties in a Design Language and its execution environment – Industrial Fellowship (CIFRE / Thales)

Participants: Charles Consel, Emilie Balland, Stéphanie Gatti, Quentin Enard.

The goal of this project is to add non-functional properties in the DIASPEC language and in the DIAGEN generator. More especially, these non-functional properties are considered on three different levels:

- *The component level.* The non-functional properties define temporal, physical and software constraints restrictive for a component.
- *The component coupling level.* The non-functional properties define the dependency between the components as well as the Quality of Service provided and required by each component of the environment.
- *The software architecture level.* The non-functional properties describe the resources that must be allocated to a component (memory, processing capacity). They also define the necessary resources for a component to interact with other components (network QoS).

This work will be illustrated and validated with a concrete application in the avionics domain.

8. Partnerships and Cooperations

8.1. Regional Initiatives

- Assistive Technologies for Elderly

  The objective of this project is to provide an open platform of digital assistance dedicated to aging in place. This project is in collaboration with researchers in Cognitive Science (Bordeaux University) and the UDCCAS Gironde (Union Départementale des Centres Communaux d'Action Sociale) managing elderly care. This project will include a need analysis, the development of new assistive applications and their experimental validation.

  This work is funded by CARSAT Aquitaine ("Caisse d'Assurance Retraite et de la Santé au Travail").
- Cognitive Assistance for Supporting the Autonomy of Persons with Intellectual Disabilities

  The objective of this project is to develop assistive technologies enabling people with intellectual disabilities to gain independence and to develop self-determined behaviors, such as making choices and taking decisions. This project is in collaboration with the "Handicap et Système Nerveux" reserach group (EA 4136, Bordeaux University), the TSA Chair of UQTR (Université du Québec à Trois-Rivières) in Psychology and the Association Trisomie 21 Gironde (Down's Syndrom). The TSA chair has recently designed and built a smart apartment that is used to conduct experimental evaluation of our assistive technologies in realistic conditions.

8.2. National Initiatives

- SmartImmo: Towards intelligent and environmentally-friendly buildings

  The SmartImmo project gathers research groups in pervasive systems and french companies working in the building construction, installation, and management. This project led by Orange Labs aims to make a building able to "communicate" with its occupants and to be environmentally-friendly (*e.g.,* automatic temperature adjusting). The main objectives of this project are to design a M2M (Machine-To-Machine) box for the heterogeneous equipment communication and to build several services on top of this platform.

  This project is funded by the SCS (Secured Communicating Solutions), a french pole of competitiveness.

- SERUS: Software Engineering for Resilient Ubiquitous Systems

  The objectives of this project is to propose a design-driven development methodology for resilient systems that takes into account dependability concerns in the early stages, ensures the traceability of these requirements throughout the system life-cycle, even during runtime evolution. To provide a high level of support, this methodology will rely on a design paradigm dedicated to sense/compute/control applications. This design will be enriched with dependability requirements and used to provide support throughout the system life-cycle. This project is in collaboration with the TSF-LAAS research group (CNRS, Toulouse) and the ADAM research project-team (Inria Lille Nord Europe).

  This work is funded by the Inria collaboration program (in French, "actions de recherches collaboratives").

- School Inclusion for Children with Autism

  The objective of this project is to provide children with assistive technologies dedicated to the school routines. This project is in collaboration with the "Handicap et Système Nerveux" research group (EA 4136, Bordeaux University), the PsyCLÉ research center (EA 3273, Provence Aix-Marseille University) and the "Parole et Langage" research laboratory (CNRS, Provence Aix-Marseille University).

  This work is funded by the French Ministry of National Education.

8.3. European Initiatives

8.3.1. Collaborations in European Programs, except FP7

Program: SUDOE territorial cooperation program (Interreg IV B)

Project acronym: Biomasud

Project title: Mechanisms for sustainability and enhancement of solid biomass market in the space of SUDOE

Duration: July 2011 - June 2013

Coordinator: AVEBIOM

Other partners: UCE (Consumers Union of Spain), CIEMAT (Public Research Agency for excellence in energy and environment, Spain), CBE (Centro da Biomassa para a Energia, Portugal), CVR (Centro para la Valorización de Residuos, Portugal) and UCFF (Union Française de la Coopération Forestière, France)

Abstract: The goal of the Biomasud european project is to show the viability of the biomass-based energy model. The project aims to propose a certification and traceability process throughout the value chain of biofuel. Our objective is to design and implement a prototype of traceability system that will extract automatically traceability information based on sensors such as RFID tags, simplifying the certification process. This work will leverage our DIASUITE development methodology and will be evaluated by the Biomasud partners.

8.3.2. Major European Organizations with which you have followed Collaborations

University of Copenhagen, DIKU (Denmark)

Subject: we have been exchanging visits and publishing articles with Julia Lawall

8.4. International Initiatives

8.4.1. Inria International Partners

- University of McGill, Montréal, Canada
- University of Québec, Trois-Rivières, Canada

8.4.2. Visits of International Scientists

The Phoenix group has been visited by:

- Scott Lee (University of Auckland, New Zealand) on April 27, 2011.
- Kay Connely (Indiana University, US) from October 6, 2011 to October 7, 2011.
- Dany Lussier-Desrochers (University of Québec, Trois-Rivières, Canada) from October 3, 2011 to October 7, 2011.

9. Dissemination

9.1. Animation of the scientific community

Charles Consel has been involved in the following events as:

- Program Committee member of
  - ICWS 2011: IEEE International Conference on Web Services
  - CyPhy 2011: Workshop on Design, Modeling and Evaluation of Cyber Physical Systems
  - EUC 2011: IEEE/IFIP International Conference on Embedded and Ubiquitous Computing
- Invited speaker at
  - NOTERE 2011: 11th annual International Conference on New Technologies of Distributed Systems
  - PEPM 2011: ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation
  - INS2I Scientific Council (invited talk on ambient computing) on September 29, 2011
  - Boulder University, Colorado, July 2011
- Guest Editor for the Annals of Telecommunications, Springer
- Member of the scientific commitee on “GDR génie de la programmation du logiciel” (CNRS)
- Member of the steering committee of the International Conference on Generative Programming and Component Engineering (GPCE)
- Member of the IFIP WG 2.11 on Program Generation
- Member of the Inria working group on research and perspectives in the domain “Réseaux, systèmes et services, calcul distribué”
- French representative for IFIP TC2 "Software Theory and Practice"
- Member of the AERES evaluation committee of the I3S laboratory (UMR 6070), Sophia-Antipolis

Charles Consel has participated in the following thesis defense committees:

- Henner Jakob, June 27, University of Bordeaux, France
- Damien Cassou, March 17, University of Bordeaux, France
- Julien Mercadal, October 10, University of Bordeaux, France
- Adrienne Tankeu Choitat, December 16, University of Toulouse, France

Emilie Balland has been involved as in the following events as:

- Reviewer of the journal “Science of Computer Programming”
- Program Committee member of
  - WASDeTT 2011: 4th International Workshop on Academic Software Development Tools and Techniques
  - LDTA 2011: 11th Workshop on Language Descriptions, Tools and Applications (Co-located with ETAPS 2011)
  - DSL 2011: 2th IFIP Working Conference on Domain Specific Languages
- Local chair of
  - LDTA 2011: 11th Workshop on Language Descriptions, Tools and Applications
  - DSL 2011: 2th IFIP Working Conference on Domain Specific Languages
- Member of the Inria local committee on Popular Science Events
- Member of the IFIP WG 2.11 on Program Generation

Participation of the Phoenix Inria project team in the following events:

- "Les Rencontres du Numérique éducatif et culturel" on assistive technology for autistic children at school, March 2011
- “Fête de la science” (national popular science event), October 2011
- “Journées Eurêka” (regional popular science event), October 2011
- Rencontre Inria Idustries “Les sciences du numérique au service de la santé à domicile et de l’autonomie”, October 2011
- Signature of the convention on disability with FIPHFP (Fonds pour l’Insertion des Personnes Handicapées dans la Fonction Publique) by Inria, November 2011

9.2. Teaching

Teaching:

Master: Domain-Specific Languages, Charles Consel, 13 hours (M2 level), ENSEIRB engineering school, France.

Master: Telephony over IP, Charles Consel, 10 hours (M2 level), ENSEIRB engineering school, France.

Master: Domain-Specific Languages for Telephony Services, Charles Consel, 15 hours (M2 level), ENSEIRB engineering school, France.

Master: Architecture Description Languages, Charles Consel, 12 hours (M2 level), ENSEIRB engineering school, France.

Master: Software Development guided by modeling and verification, Emilie Balland, 20 hours (M2 level), ENSEIRB engineering school, France.

PhD & HdR:

PhD : Henner Jakob, “Vers la sécurisation des systèmes d’informatique ubiquitaire par le design : une approche langage”, University of Bordeaux, June 27, 2011, supervised by Charles Consel

PhD : Damien Cassou, “Développement logiciel orienté paradigme de conception : la programmation dirigée par la spécification”, University of Bordeaux, March 17, 2011, supervised by Charles Consel

PhD : Julien Mercadal, “Approche langage au développement logiciel : application au domaine des systèmes d’informatique ubiquitaire”, University of Bordeaux, October 10, 2011, supervised by Charles Consel

PhD in progress : Julien Bruneau, “Plateforme d’exécution paramétrable de systèmes communicants”, October 2008, supervised by Charles Consel

PhD in progress : Hongyu Guan, “Gestion de l’hétérogénéité des environnements ubiquitaires et de la consommation d’énergie des environnements mobiles”, started in February 2009, supervised by Charles Consel

PhD in progress : Pengfei Liu, “Politiques de securité pour les environnements ubiquitaires”, started in October 2009, supervised by Charles Consel

PhD in progress : Stéphanie Gatti, “Architecture en composants et qualification incrémentale”, started in February 2010, supervised by Charles Consel and Emilie Balland

PhD in progress : Quentin Enard, “Intégration de concepts de sûreté de fonctionnement dans un langage de description d’architecture et son support d’exécution”, started in February 2010, supervised by Charles Consel

PhD in progress : Luc Vercellin, “Prise en compte des concepts d’adaptation dans le développement d’applications d’assistance cognitive”, started in October 2011, supervised by Charles Consel and Emilie Balland

10. Bibliography

Major publications by the team in recent years

[1] D. CASSOU, E. BALLAND, C. CONSEL, J. LAWALL. *Leveraging Software Architectures to Guide and Verify the Development of Sense/Compute/Control Applications*, in "ICSE’11: Proceedings of the 33rd International Conference on Software Engineering", Honolulu, United States, ACM, 2011, p. 431-440, http://hal.inria.fr/inria-00537789/en.

[2] D. CASSOU, J. BRUNEAU, C. CONSEL, E. BALLAND. *Towards a Tool-based Development Methodology for Pervasive Computing Applications*, in "IEEE Transactions on Software Engineering", October 2011, http://hal.inria.fr/inria-00631477/en.

[3] C. CONSEL. *From A Program Family To A Domain-Specific Language*, Lecture Notes in Computer Science, State-of-the-Art Survey, Springer-Verlag, 2004, n^{o} 3016, p. 19–29, http://phoenix.labri.fr/publications/papers/dagstuhl-consel.pdf.

[4] C. CONSEL, J. LAWALL, A.-F. LE MEUR. *A Tour of Tempo: A Program Specializer for the C Language*, in "Science of Computer Programming", 2004, http://phoenix.labri.fr/publications/papers/tour-tempo.ps.gz.

[5] C. CONSEL, L. RÉVEILLÈRE. *A Programmable Client-Server Model: Robust Extensibility via DSLs*, in "Proceedings of the 18th IEEE International Conference on Automated Software Engineering (ASE 2003)", Montréal, Canada, IEEE Computer Society Press, November 2003, p. 70–79, http://phoenix.labri.fr/publications/papers/Consel-Reveillere_ase03.pdf.

[6] C. CONSEL, L. RÉVEILLÈRE. *A DSL Paradigm for Domains of Services: A Study of Communication Services*, Lecture Notes in Computer Science, State-of-the-Art Survey, Springer-Verlag, 2004, n° 3016, p. 165–179, http://phoenix.labri.fr/publications/papers/dagstuhl04_consel_reveillere.pdf.

[7] A.-F. LE MEUR, J. LAWALL, C. CONSEL. *Specialization Scenarios: A Pragmatic Approach to Declaring Program Specialization*, in "Higher-Order and Symbolic Computation", 2004, vol. 17, n° 1, p. 47–92, http://phoenix.labri.fr/publications/papers/spec-scenarios-hosc2003.ps.gz.

[8] D. MCNAMEE, J. WALPOLE, C. PU, C. COWAN, C. KRASIC, A. GOEL, P. WAGLE, C. CONSEL, G. MULLER, R. MARLET. *Specialization tools and techniques for systematic optimization of system software*, in "ACM Transactions on Computer Systems", May 2001, vol. 19, n° 2, p. 217–251, http://phoenix.labri.fr/publications/papers/tocs01-namee.pdf.

[9] J. MERCADAL, Q. ENARD, C. CONSEL, N. LORIANT. *A Domain-Specific Approach to Architecturing Error Handling in Pervasive Computing*, in "OOPSLA'10: Proceedings of the 25th Annual ACM SIGPLAN Conference on Object Oriented Programming Systems Languages and Applications", États-Unis Reno, October 2010, http://hal.inria.fr/inria-00486930/en.

[10] F. MÉRILLON, L. RÉVEILLÈRE, C. CONSEL, R. MARLET, G. MULLER. *Devil: An IDL for Hardware Programming*, in "Proceedings of the Fourth Symposium on Operating Systems Design and Implementation", San Diego, California, October 2000, p. 17–30, http://phoenix.labri.fr/publications/papers/osdi00-merillon.pdf.

Publications of the year

Doctoral Dissertations and Habilitation Theses

[11] D. CASSOU. *Développement logiciel orienté paradigme de conception : la programmation dirigée par la spécification*, Université Sciences et Technologies - Bordeaux I, March 2011, http://hal.inria.fr/tel-00583246/en.

[12] H. JAKOB. *Vers la sécurisation des systèmes d'informatique ubiquitaire par le design : une approche langage*, Université Sciences et Technologies - Bordeaux I, June 2011.

[13] J. MERCADAL. *Approche langage au développement logiciel : application au domaine des systèmes d'informatique ubiquitaire*, Université Sciences et Technologies - Bordeaux I, October 2011.

Articles in International Peer-Reviewed Journal

[14] D. CASSOU, J. BRUNEAU, C. CONSEL, E. BALLAND. *Towards a Tool-based Development Methodology for Pervasive Computing Applications*, in "IEEE Transactions on Software Engineering", October 2011, http://hal.inria.fr/inria-00631477/en.

International Conferences with Proceedings

[15] D. CASSOU, E. BALLAND, C. CONSEL, J. LAWALL. *Leveraging Software Architectures to Guide and Verify the Development of Sense/Compute/Control Applications*, in "ICSE'11: Proceedings of the 33rd International Conference on Software Engineering", Honolulu, United States, ACM, 2011, p. 431-440, http://hal.inria.fr/inria-00537789/en.

[16] C. CONSEL. *DiaSuite:A Paradigm-Oriented Software Development Approach (invited paper)*, in "20th ACM SIGPLAN workshop on Partial evaluation and program manipulation : PEPM'11", Austin, TX, United States, ACM, January 2011, p. 77-78 [*DOI :* 10.1145/1929501.1929515], http://hal.inria.fr/inria-00581652/en.

[17] S. GATTI, E. BALLAND, C. CONSEL. *A Step-wise Approach for Integrating QoS throughout Software Development*, in "FASE'11: Proceedings of the 14th European Conference on Fundamental Approaches to Software Engineering", Sarrebruck, Germany, Lecture Notes in Computer Science, Springer, March 2011, vol. 6603, p. 217-231, http://hal.inria.fr/inria-00561619/en.

[18] H. JAKOB, C. CONSEL, N. LORIANT. *Architecturing Conflict Handling of Pervasive Computing Resources*, in "11th IFIP International Conference on Distributed Applications and Interoperable Systems", Reykjavik, Iceland, Lecture Notes in Computer Science, Springer, June 2011, vol. 6723, p. 92-105, http://hal.inria.fr/inria-00581604/en.

National Conferences with Proceeding

[19] D. CASSOU, C. CONSEL, E. BALLAND, J. LAWALL. *Faire levier sur les architectures logicielles pour guider et vérifier le développement d'applications SCC*, in "GDR GPL'11: 3ème journées du Génie de la programmation et du logiciel", Lille, France, June 2011, p. 33–34, http://hal.inria.fr/inria-00602098/en.

References in notes

[20] P. BOINOT, R. MARLET, J. NOYÉ, G. MULLER, C. CONSEL. *A Declarative Approach for Designing and Developing Adaptive Components*, in "Proceedings of the 15th IEEE International Conference on Automated Software Engineering (ASE 2000)", Grenoble, France, IEEE Computer Society Press, September 2000, p. 111–119.

[21] S. CHIROKOFF, C. CONSEL, R. MARLET. *Combining Program and Data Specialization*, in "Higher-Order and Symbolic Computation", December 1999, vol. 12, n^{o} 4, p. 309–335.

[22] C. CONSEL, J. LAWALL, A.-F. LE MEUR. *A Tour of Tempo: A Program Specializer for the C Language*, in "Science of Computer Programming", 2004.

[23] C. CONSEL, R. MARLET. *Architecturing software using a methodology for language development*, in "Proceedings of the 10th International Symposium on Programming Language Implementation and Logic Programming", Pisa, Italy, C. PALAMIDESSI, H. GLASER, K. MEINKE (editors), Lecture Notes in Computer Science, September 1998, vol. 1490, p. 170–194.

[24] A. K. DEY, G. D. ABOWD, D. SALBER. *A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications*, in "Human-Computer Interaction", 2001, vol. 16, n^{o} 2, p. 97–166.

[25] A.-F. LE MEUR, C. CONSEL, B. ESCRIG. *An Environment for Building Customizable Software Components*, in "IFIP/ACM Conference on Component Deployment", Berlin, Germany, June 2002, p. 1–14.

[26] A.-F. LE MEUR, C. CONSEL. *Generic Software Component Configuration Via Partial Evaluation*, in "SPLC'2000 Workshop – Product Line Architecture", Denver, Colorado, August 2000.

[27] A.-F. LE MEUR, J. LAWALL, C. CONSEL. *Towards Bridging the Gap Between Programming Languages and Partial Evaluation*, in "ACM SIGPLAN Workshop on Partial Evaluation and Semantics-Based Program Manipulation", Portland, OR, USA, ACM Press, January 2002, p. 9–18.

[28] A.-F. LE MEUR, J. LAWALL, C. CONSEL. *Specialization Scenarios: A Pragmatic Approach to Declaring Program Specialization*, in "Higher-Order and Symbolic Computation", 2004, vol. 17, n^{o} 1, p. 47–92.

[29] R. MARLET, C. CONSEL, P. BOINOT. *Efficient Incremental Run-Time Specialization for Free*, in "Proceedings of the ACM SIGPLAN'99 Conference on Programming Language Design and Implementation (PLDI'99)", Atlanta, GA, USA, May 1999, p. 281–292.

[30] R. MARLET, S. THIBAULT, C. CONSEL. *Mapping Software Architectures to Efficient Implementations via Partial Evaluation*, in "Conference on Automated Software Engineering", Lake Tahoe, NV, USA, IEEE Computer Society, November 1997, p. 183–192.

[31] R. MARLET, S. THIBAULT, C. CONSEL. *Efficient Implementations of Software Architectures via Partial Evaluation*, in "Journal of Automated Software Engineering", October 1999, vol. 6, n^{o} 4, p. 411–440.

[32] D. MCNAMEE, J. WALPOLE, C. PU, C. COWAN, C. KRASIC, A. GOEL, P. WAGLE, C. CONSEL, G. MULLER, R. MARLET. *Specialization tools and techniques for systematic optimization of system software*, in "ACM Transactions on Computer Systems", May 2001, vol. 19, n^{o} 2, p. 217–251.

[33] N. MEDVIDOVIC, R. N. TAYLOR. *A Classification and Comparison Framework for Software Architecture Description Languages*, in "IEEE Transactions on Software Engineering", 2000, vol. 26, n^{o} 1, p. 70–93, http://dx.doi.org/10.1109/32.825767.

[34] G. MULLER, C. CONSEL, R. MARLET, L. BARRETO, F. MÉRILLON, L. RÉVEILLÈRE. *Towards Robust OSes for Appliances: A New Approach Based on Domain-Specific Languages*, in "Proceedings of the ACM SIGOPS European Workshop 2000 (EW2000)", Kolding, Denmark, ACM Press, September 2000, p. 19-24.

[35] F. MÉRILLON, L. RÉVEILLÈRE, C. CONSEL, R. MARLET, G. MULLER. *Devil: An IDL for Hardware Programming*, in "4th Symposium on Operating Systems Design and Implementation (OSDI 2000)", San Diego, California, October 2000, p. 17–30.

[36] L. RÉVEILLÈRE, F. MÉRILLON, C. CONSEL, R. MARLET, G. MULLER. *A DSL Approach to Improve Productivity and Safety in Device Drivers Development*, in "Proceedings of the 15th IEEE International Conference on Automated Software Engineering (ASE 2000)", Grenoble, France, IEEE Computer Society Press, September 2000, p. 101–109.

[37] L. RÉVEILLÈRE, G. MULLER. *Improving Driver Robustness: an Evaluation of the Devil Approach*, in "The International Conference on Dependable Systems and Networks", Göteborg, Sweden, IEEE Computer Society, July 2001, p. 131–140.

[38] U. SCHULTZ, J. LAWALL, C. CONSEL, G. MULLER. *Towards Automatic Specialization of Java Programs*, in "Proceedings of the European Conference on Object-oriented Programming (ECOOP'99)", Lisbon, Portugal, Lecture Notes in Computer Science, June 1999, vol. 1628, p. 367–390.

[39] U. SCHULTZ, J. LAWALL, C. CONSEL. *Specialization Patterns*, in "Proceedings of the 15th IEEE International Conference on Automated Software Engineering (ASE 2000)", Grenoble, France, IEEE Computer Society Press, September 2000, p. 197–208.

[40] U. SCHULTZ, J. LAWALL, C. CONSEL. *Automatic Program Specialization for Java*, in "ACM Transactions on Programming Languages and Systems", 2003, vol. 25, n^{o} 4, p. 452–499.

[41] S. THIBAULT, C. CONSEL, J. LAWALL, R. MARLET, G. MULLER. *Static and Dynamic Program Compilation by Interpreter Specialization*, in "Higher-Order and Symbolic Computation", September 2000, vol. 13, n^{o} 3, p. 161–178.

[42] S. THIBAULT, R. MARLET, C. CONSEL. *Domain-Specific Languages: from Design to Implementation – Application to Video Device Drivers Generation*, in "IEEE Transactions on Software Engineering", May 1999, vol. 25, n^{o} 3, p. 363–377.