



INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Phoenix

*Programming Language Technology For
Communication Services*

Futurs

THEME COM

Activity
R *eport*
2007

Table of contents

1. Team	1
2. Overall Objectives	1
2.1. Context	1
2.2. Overview	2
2.3. Highlights of the year	2
3. Scientific Foundations	2
3.1. Introduction	2
3.2. Adaptation Methodologies	3
3.2.1. Domain-Specific languages	3
3.2.2. Declaring adaptation	3
3.2.3. Declaring specialization	3
3.2.4. Specializing design patterns	3
3.2.5. Specializing software architectures	4
3.3. Adaptation in Systems Software	4
3.3.1. DSLs in Operating Systems	4
3.3.2. Devil - a DSL for device drivers	4
3.3.3. Plan-P - a DSL for programmable routers	4
3.4. Adaptation Tools and Techniques	5
4. Application Domains	5
4.1. Telephony Services	5
4.2. Multimedia Streaming Services	6
5. Software	6
5.1. Tempo - A Partial Evaluator for C	6
5.2. SPL - A Domain-Specific Language for Robust Session Processing Services	7
5.3. Stingy - A Domain-Specific Compiler for High-performance Network Servers	7
5.4. VisuCom - A Graphical Telephony Service Creation Environment and its Application Server	7
5.5. Patent	8
5.6. Zebu - A Domain Specific Language for Implementing Network Application Protocols	8
5.7. Pervasive Computing: Middleware and DSL	8
5.7.1. PerGen - A Domain-Specific IDL and its Compiler for Pervasive Computing Applications	8
5.7.2. Pantachou - A Domain-Specific Language for Developing Safe Coordination Services	8
6. New Results	8
6.1. Remote Specialization for Efficient Embedded Operating Systems	8
6.2. A Language-Based Approach for Improving the Robustness of Network Application Protocol Implementations	9
6.3. A Stepwise Approach to Developing Languages for SIP Telephony Service Creation	9
6.4. Building Home Monitoring Applications: From Design to Implementation into The Amigo Middleware	10
6.5. Staging Telephony Service Creation: A Language Approach	10
6.6. Ontology-Directed Generation of Frameworks For Pervasive Service Development	10
6.7. A Domain-Specific IDL and its Compiler for Pervasive Computing Applications	10
7. Contracts and Grants with Industry	11
7.1. ACI Security COrSS	11
7.2. Ambient Intelligence For The Networked Home Environment (IP6 Amigo)	11
7.3. Service Oriented Architecture for Embedded Systems – Industrial Fellowship (CIFRE / Thales)	12
7.4. Capability-based DSLs – Région Aquitaine Fellowship	12

7.5. Designing techniques and tools for developing domain-specific languages – Industrial Fellowship (CIFRE / Thales)	12
7.6. Language Families for Systems Families (ANR Blanc)	12
7.7. HomeSIP: development of a SIP-based middleware for home automation (France Telecom)	13
8. Other Grants and Activities	13
8.1. International Collaborations	13
8.2. Visits and Invited Researchers	13
9. Dissemination	14
9.1. Scientific Community Participation	14
9.2. Teaching	14
9.3. Presentations and Invitations	15
10. Bibliography	15

1. Team

The Phoenix group is located in Bordeaux. Phoenix is a joint research group with LaBRI (Laboratoire Bordelais de Recherche en Informatique) – the computer science department at the University of Bordeaux I – CNRS (Centre National de la Recherche Scientifique) – a French national scientific research center – and ENSEIRB (Ecole Nationale Supérieure en Electronique, Informatique et Radiocommunications de Bordeaux) – an electronics, computer science, and telecommunications engineering school at Bordeaux. The group is located at the INRIA-Futurs facility in Bordeaux.

Head of project-team

Charles Consel [Professor, ENSEIRB, HdR]

Research scientist

Laurent Réveillère [Associate Professor, ENSEIRB]

Administrative assistant

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

External collaborators

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

Post-doctoral fellows

Yérom-David Bromberg [Inria scholarship, from December 4, 2006]

PhD students

Laurent Burgy [Inria scholarship, from October 1, 2004, University of Bordeaux 1]

Fabien Latry [Inria scholarship, from October 1, 2004, to September 21, 2007, University of Bordeaux 1]

Nicolas Palix [Inria scholarship, from October 1, 2004, University of Bordeaux 1]

Wilfried Jouve [Inria scholarship, from October 3, 2005, University of Bordeaux 1]

Julien Lancia [Thales (industrial Ph.D. student), from November 14, 2005, University of Bordeaux 1]

Julien Mercadal [Ministerial scholarship, from October 2, 2006, University of Bordeaux 1]

Zoé Drey [Thales (industrial Ph.D. student), from November 2, 2006, University of Bordeaux 1]

Damien Cassou [Ministerial scholarship, from October 1, 2007, University of Bordeaux 1]

Technical staff

Thomas Rougelot [Associate Engineer, from September 4, 2006]

Benjamin Bertran [Associate Engineer, from December 17, 2007]

Fabien Latry [Research Engineer, from September 22, 2007]

Student interns

Julien Bruneau [Second-year student, from June 1, 2007, to September 30, 2007, ENSEIRB]

Former PHD Students

Sapan Bhatia [Post-doctoral fellows, University of Princeton]

Fabien Latry [Research Engineer]

2. Overall Objectives

2.1. Context

Keywords: *Operating systems, client-server model, communication services, compilation, domain analysis and engineering, language design, networking, program analysis and transformation, specialization, telephony.*

The frantic nature of technological advances in the area of multimedia communications, compounded with the effective convergence between telecommunication and computer networks, is opening up a host of new functionalities, placing service creation as a fundamental vehicle to bring these changes to end-users.

This situation has three main consequences: (1) service creation is increasingly becoming a *software intensive area*; (2) because communication services are often heavily relied on, intensive service creation must preserve *robustness*; (3) the growing multimedia nature of communication services imposes *high-performance requirements* on services and underlying layers.

2.2. Overview

Keywords: *Operating systems, client-server model, communication services, compilation, domain analysis and engineering, language design, networking, program analysis and transformation, specialization, telephony.*

The phoenix group aims to develop principles, techniques and tools for the development of *communication services*. To address the requirements of this domain, the scope of our research comprises the key elements underlying communication services: the infrastructure that enables communication to be set up (*e.g.*, signalling platform, transport protocols, and session description); the software architecture underlying services (*e.g.*, the client-server model, programming interfaces, and the notion of service logic); and, communication terminals (*e.g.*, terminal features and embedded systems).

Our approach covers three key aspects of the area of communication services: (1) definition of new Domain-Specific Languages (DSLs), using programming language technology to enable the specification of robust services; (2) study of the layers underlying communication services to improve flexibility and performance; (3) application to concrete areas to validate our approach.

2.3. Highlights of the year

A Bordeaux-based startup company, named Siderion Technologies, has been created on May 11 2007. Siderion proposes a solution targeting applicative telephony, enabling users to customize the routing of their phone calls with respect to their own rules, a contact base and various preferences. Thanks to a graphical environment, call routing services can be easily and safely created, covering a large spectrum of professional profiles and preferences. The VisuCom technology has been developed in the Phoenix research group at INRIA.

3. Scientific Foundations

3.1. Introduction

Our proposed project builds upon results that have been 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 can 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 [24]. 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 [38]. 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 [37].

3.2.5. *Specializing software architectures*

The source 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 systematically be applied to optimize those mechanism [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) [33]. This approach was based on DSLs and enables rapid development of robust OSes. Such approach is critically needed in application domain, 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 [35]. 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 [34]: 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 [36]. Mutation analysis were 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.

3.3.3. *Plan-P - a DSL for programmable routers*

Besides device drivers, the Compose group also explored the area of networking in the context of DSLs. More specifically, we developed a language, named Plan-P, that enables the network to be programmable and thus to offer extensibility [41]. As such, Plan-P enables protocols to be defined for specific applications. Plan-P extends a language, named Plan, developed by the University of Pennsylvania and devoted to network diagnostics. Plan-P enables routers to be programmed in a safe and secure way without any loss in bandwidth. To achieve safety and security, the language is restricted, and programs are downloaded into the routers as DSL source code to enable thorough verifications. For efficiency, a light Just-In-Time compiler is generated from the Plan-P interpreter via program specialization. This compiler is installed on routers to compile uploaded Plan-P source code.

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 [23]. This analysis is at the heart of our program specializer for C, named Tempo [23]. 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 [40]. 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 [37]. 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 [39].

4. Application Domains

4.1. Telephony Services

Keywords: *SIP, adaptation, multimedia, telecommunications.*

IP telephony materializes the convergence between telecommunications and computer networks. This convergence is dramatically changing the face of the telecommunications domain moving from proprietary, closed platforms to distributed systems based on network protocols. In particular, a telephony platform is based on a client-server model and consists of a *signalling server* that implements a particular signalling protocol (e.g., the Session Initiation Protocol [19]). A signalling server is able to perform telephony-related operations that include resources accessible from the computer network, such as Web resources, databases... This evolution brings a host of new functionalities to the domain of telecommunications. Such a wide spectrum of functionalities enables Telephony to be customized with respect to preferences, trends and expectations of ever demanding users. These customizations critically rely on a proliferation of telephony services. In fact,

introducing new telephony services is facilitated by the open nature of signalling servers, as shown by all kinds of servers in distributed systems. However, in the context of telecommunications, such evolution should lead service programming to be done by non-expert programmers, as opposed to developers certified by telephony manufacturers. To make this evolution worse, the existing techniques to program server extensions (*e.g.*, Common Gateway Interface [18]) are rather low level, involves crosscutting expertises (*e.g.*, networking, distributed systems, and operating systems) and requires tedious session management. These shortcomings make the programming of telephony services an error-prone process, jeopardizing the robustness of a platform.

We are developing a DSL, named SPL (*Session Processing Language*), aimed to ease the development of telephony services without sacrificing robustness.

4.2. Multimedia Streaming Services

Keywords: *adaptation, multimedia, streaming, telecommunications.*

Mobility and wireless networks pose a major challenge to media delivery: how does one mass-deliver media while at the same time personalizing it to account for diverse needs such as multiple heterogeneous rendering terminals, user requirements, network bandwidth, *etc.* Such personalization involves transcoding and transforming multimedia resources along the image chain.

To do so, various treatments, commonly supported by hardware, are gradually being shifted to software, to face unpredictable needs. On the one hand, this shift helps to keep pace with the rapidly evolving domain of media delivery. On the other hand, it imposes very high-performance requirements for treatments that were earlier hardware supported. As a consequence, developing a streaming application often involves low-level programming, critical memory management, and finely tuned scheduling of processing steps.

To address these problems, we have designed and implemented a DSL, named *Spidle*, for specifying streaming applications [22]. Our approach consists in

- Identifying (and possibly modifying) a protocol (*e.g.*, RTSP) for multimedia streaming.
- Making a streaming server, based on the previously identified protocol, programmable using *Spidle*. This work will permit streaming adaptations to the client needs and preferences.
- Defining realistic adaptation scenarios to validate our approach. This work may lead us to extend *Spidle* to cope with the target scenarios.
- Assessing our approach by conducting a thorough experimental study.

5. Software

5.1. Tempo - A Partial Evaluator for C

Keywords: *C language, partial evaluation, run-time specialization.*

Participants: Charles Consel [correspondent], Julia Lawall.

Tempo is a partial evaluator for C programs. It is an off-line specializer; it is divided into two phases: analysis and specialization.

The input to the analysis phase consists of a program and a description of which inputs will be known during specialization and which will be unknown. Based on this knowledge, dependency analyses propagate information about known and unknown values throughout the code and produce an annotated program, indicating how each program construct should be transformed during specialization. Because C is an imperative language including pointers, the analysis phase performs alias and side-effect analyses in addition to binding-time analyses. The accuracy of these analyses is targeted towards keeping track of known values across procedures, data structures, and pointers. Following the analysis phase, the specialization phase generates a specialized program based on the annotated program and the values of the known inputs. Tempo can specialize programs at compile time (*i.e.*, source-to-source transformation) as well as run time (*i.e.*, run-time binary code generation).

The Tempo specializer has been applied in various domains such as operating systems and networking, computer graphics, scientific computation, software engineering and domain specific languages. It has been made publicly available since April 1998. Its documentation is available on line, as well as tutorial slides.

5.2. SPL - A Domain-Specific Language for Robust Session Processing Services

Keywords: *SIP, adaptation, services, sessions, telephony.*

Participants: Charles Consel, Laurent Réveillère [correspondent], Laurent Burgy, Fabien Latry, Nicolas Palix.

SPL is a high-level domain-specific language for specifying robust Internet telephony services.

SPL reconciles programmability and reliability of telephony services, and offers high-level constructs that abstract over intricacies of the underlying protocols and software layers. SPL makes it possible for owners of telephony platforms to deploy third-party services without compromising safety and security. This openness is essential to have a community of service developers that addresses such a wide spectrum of new functionalities. The SPL compiler is nearing completion.

5.3. Stingy - A Domain-Specific Compiler for High-performance Network Servers

Keywords: *Cache Optimizations, Domain-specific optimizations, Event-driven Programs.*

Participants: Sapan Bhatia [correspondent], Charles Consel, Julia Lawall.

Event-driven programming has emerged as a standard to implement high-performance servers due to its flexibility and low OS overhead. Still, memory access remains a bottleneck. Generic optimization techniques yield only small improvements in the memory access behavior of event-driven servers, as such techniques do not exploit their specific structure and behavior.

The Stingy compiler implements an optimization framework dedicated to event-driven servers, based on a strategy to eliminate data-cache misses. Our approach exploits the flexible scheduling and deterministic execution of event-driven servers. It is based on a novel memory manager combined with a tailored scheduling strategy to restrict the working data set of the program to a memory region mapped directly into the data cache.

In practice, the Stingy compiler accepts as input an event-driven server written in C and annotated to expose a specific memory management and scheduling interface. As output, it generates C code for an optimized version of the server. The Stingy compiler has been tested on the following servers: The TUX, tthtpd, Flash, boa, mathopd. It has also been applied to the Cactus QoS framework and the Squid proxy server. The highest speedup observed under heavy loads is on the TUX server (in the range of 40%). For the remaining servers, gains are in the region of 10-15%.

5.4. VisuCom - A Graphical Telephony Service Creation Environment and its Application Server

Keywords: *SIP, semantic verifications, services creation, telephony.*

Participants: Charles Consel, Laurent Réveillère, Laurent Burgy, Fabien Latry [correspondent], Nicolas Palix.

To ease the development of telephony services, a DSL known as SPL (Session Processing Language) has been designed and implemented. This language offers domain-specific constructs and extensions that abstract over the intricacies of the underlying technologies. To enable non-programmers to define services, a graphical telephony service creation workshop, named VisuCom, has been developed, as well as its corresponding execution environment. This software offers intuitive visual constructs and menus that permit users to quickly develop a wide variety of services ranging from simple redirection to agenda dependent call handling. The whole architecture also enables semantic properties to be verified. Examples of errors detected in services include call loss, incorrect state transitions, and unbounded resource usage.

VisuCom and its execution environment are deposited at the Agency for the Protection of Programs (APP).

5.5. Patent

Participants: Laurent Burgy, Charles Consel, Fabien Latry, Nicolas Palix, Laurent Réveillère.

"Dispositif d'interconnexion dun système dinformations dentreprise(s) à un serveur", Inria patent No. 06291276.1, August 7, 2006.

5.6. Zebu - A Domain Specific Language for Implementing Network Application Protocols

Keywords: *DSL, Message processing, Network protocols.*

Participants: Laurent Burgy [correspondent], Laurent Réveillère.

In the Internet era, many applications, ranging from instant messaging clients and multimedia players to HTTP servers and proxies, involve processing network protocol messages. However, implementing a correct and efficient network protocol message parser is a difficult task. To address this issue, we have recently introduced a declarative language, Zebu, for describing protocol message formats and related processing constraints. We have found that Zebu-based applications are often as efficient as hand-crafted ones while offering more safety and robustness.

5.7. Pervasive Computing: Middleware and DSL

Keywords: *DSL, SIP, middleware, pervasive.*

Nowadays, a lot of buildings contain an amazing number of devices that have various technological functionalities. Some of them can be seen as rendering devices, like TV monitors and speakers; others are data-collecting devices like webcams and sensors; a third category is activation-based devices like lights and doors. Of course, some devices may combine more than one capability. Developing a smart building critically relies on the ability to combine the capabilities of the available devices. However, because of the heterogeneity of the devices, their combination is often achieved by using ad hoc design and implementation approaches. As a consequence, the resulting platforms are usually closed and limited, preventing usage scenarios to evolve and impeding creativity.

5.7.1. PerGen - A Domain-Specific IDL and its Compiler for Pervasive Computing Applications

Participants: Damien Cassou, Charles Consel, Wilfried Jouve [correspondent], Julien Lancia, Nicolas Palix.

The goal of the Smart Space project at the INRIA facility in Bordeaux is to create a platform where a number of devices are deployed and operated via a SIP backbone. Devices are wrapped to turn them into SIP entities. Various software development approaches are being studied to design and implement a variety of pervasive computing scenarios. We are investigating the design and development of a domain-specific IDL and its compiler, dedicated to the development of pervasive computing applications.

5.7.2. Pantachou - A Domain-Specific Language for Developing Safe Coordination Services

Participants: Charles Consel, Julien Mercadal, Nicolas Palix [correspondent].

We are designing and implementing a domain-specific language, called Pantachou, dedicated to the development of coordination services for networked entities.

6. New Results

6.1. Remote Specialization for Efficient Embedded Operating Systems

Participants: Sapan Bhatia, Charles Consel.

Prior to their deployment on an embedded system, Operating Systems are commonly tailored to reduce code size and improve run-time performance. Program specialization is a promising match for this process: it is predictable, modular and allows the reuse of previously implemented specializations. A specialization engine for embedded systems must overcome three main obstacles: (i) Reusing existing compilers for embedded systems, (ii) supporting specialization on a resource-limited system and (iii) allowing applications to invoke specialization of system code on demand.

We describe a run-time specialization infrastructure that addresses each of the above three problems. Our solution proposes: (i) Specialization in two phases of which the former generates specialized C templates and the latter uses a dedicated compiler to generate efficient native code. (ii) A virtualization mechanism that facilitates specialization of code at a remote location. (iii) Specific OS extensions that allow applications to produce, manage and dispose off specialized code.

We evaluate our work through two case studies: (i) The TCP/IP implementation of Linux and (ii) The TUX embedded web server. We report appreciable improvements in code size and performance. We also quantify the overhead of specialization and argue that a specialization server can scale to support a sizable workload. For more information, see: [11].

6.2. A Language-Based Approach for Improving the Robustness of Network Application Protocol Implementations

Participants: Laurent Burgy, Julia Lawall, Laurent Réveillère.

The secure and robust functioning of a network relies on the defect-free implementation of network applications. As network protocols have become increasingly complex, however, hand-writing network message processing code has become increasingly error-prone.

We present a domain-specific language, Zebu, for generating robust and efficient message processing layers. A Zebu specification, based on the notation used in RFCs, describes protocol message formats and related processing constraints. Zebu-based applications are efficient, since message fragments can be specified to be processed on demand. Zebu-based applications are also robust, as the Zebu compiler automatically checks specification consistency and generates parsing stubs that include validation of the message structure. Using a message torture suite in the context of SIP and RTSP, we show that Zebu-generated code is both correct and defect-free. For more information, see: [12].

6.3. A Stepwise Approach to Developing Languages for SIP Telephony Service Creation

Participants: Charles Consel, Julia Lawall, Nicolas Palix, Laurent Réveillère.

Developing a SIP-based telephony service requires a programmer to have expertise in telephony rules and constraints, the SIP protocol, distributed systems, and a SIP API, which is often large and complex. These requirements make the development of telephony software an overwhelming challenge. To overcome this challenge, various programming languages have been proposed to develop telephony services. Nevertheless, none of these languages as yet has a formal semantics. Therefore, the reference implementation, which may not be available, becomes the only source of information for the programmer to understand the subtleties of the language. Furthermore, this situation makes it difficult for third-party developers to port the language to another runtime system or to provide another implementation of the runtime system.

We present a semantics-based stepwise approach for designing and developing a scripting language dedicated to the development of telephony services. This approach enables critical properties of services to be guaranteed and captures expertise on the operational behavior of a service. We have applied this approach to developing the *Session Processing Language* (SPL) dedicated to SIP-based service creation. A variety of services have been written in SPL for our university department. For more information, see: [14].

6.4. Building Home Monitoring Applications: From Design to Implementation into The Amigo Middleware

Participants: Charles Consel, Wilfried Jouve, Laurent Réveillère.

The proliferation of smart communication devices based on technologies such as Radio-Frequency IDentification (RFID) makes pervasive computing evolving at a frantic pace. This evolution leads to the development of applications, called monitoring applications, that offer a host of new functionalities based on context information provided by tagged entities.

We introduce a software architecture dedicated to monitoring applications and define a mapping to the Amigo middleware which is dedicated to ambient computing. We illustrate our approach by developing two real-size applications for child monitoring and object reminder. In these experiments, our approach have demonstrated its usability and ease of programming. For more information, see: [15].

6.5. Staging Telephony Service Creation: A Language Approach

Participants: Fabien Latry, Julien Mercadal, Charles Consel.

The open-endedness of telephony platforms is creating expectations among users, ranging from end-users to administrators, to create services dedicated to their activities. Not only is the population of developers heterogeneous, but the technologies underlying modern telephony range over a variety of areas such as multimedia, databases, web services, and distributed systems. This situation drastically widens the expertise required for service creation.

We propose an approach to coping with the heterogeneity of both the service developers and the technologies underlying modern telephony. Our approach is based on programming languages. It consists of providing a language that is specific to each developer community with respect to its expertise (*e.g.*, programming skills) and the target application area (*e.g.*, administration). Such languages, called Domain-Specific Languages (DSLs), are organized in layers, accounting for abstraction levels.

Our layered approach to telephony service creation is illustrated by two high-level DSLs for end-user service creation, requiring no programming skills, and an expressive DSL enabling the development of expert-level telephony services. We show that layering DSLs greatly facilitates their implementation and verification of telephony-specific properties by leveraging on high-level tools. For more information, see: [16].

6.6. Ontology-Directed Generation of Frameworks For Pervasive Service Development

Participants: Charles Consel, Wilfried Jouve, Julien Lancia, Nicolas Palix.

Pervasive computing applications are tedious to develop because they combine a number of problems ranging from device heterogeneity, to middleware constraints, to lack of programming support.

We present an approach to integrating the ontological description of a pervasive computing environment into a programming language, namely Java. From this ontological description of a pervasive computing environment, a framework is automatically generated. It provides the developer with dedicated programming support to manage, discover and invoke services. Besides, it performs a number of verifications both at compile and run time, ensuring the robustness of applications. For more information, see: [13].

6.7. A Domain-Specific IDL and its Compiler for Pervasive Computing Applications

Participants: Charles Consel, Wilfried Jouve, Julien Lancia, Julia Lawall, Nicolas Palix.

Pervasive computing environments introduce new challenges for application development, due to the heterogeneity of the devices involved. In practice, pervasive computing applications rely on general-purpose middleware to manage this heterogeneity, but this approach does not provide programming support and verifications specific to the pervasive computing environment.

We present a domain-specific IDL and its compiler, dedicated to the development of pervasive computing applications. Our IDL is based on that of CORBA and provides declarative support for concisely characterizing a pervasive computing environment. This description is (1) to be used by programmers as a high-level reference to develop applications that coordinate entities of the target environment and (2) to be passed to a compiler that generates a framework dedicated to the target environment. This process enables verifications to be performed prior to runtime on both the declared environment and a given application. Furthermore, customized operations are automatically generated to support the development of pervasive computing activities, such as service discovery and session negotiation for stream-oriented devices.

We have implemented a framework generator and have used it to generate frameworks targeting pervasive computing areas such as building surveillance, advanced telecommunications and home automation. For more information, see: [17]¹.

7. Contracts and Grants with Industry

7.1. ACI Security COrSS

Participants: Laurent Burgy, Charles Consel, Fabien Latry, Nicolas Palix, Laurent Réveillère.

This project, entitled “Composition and refinement of Secure Systems”, is a collaboration between groups from the systems and formal methods community.

The goal is to study methods and tools for the development of secure and safe systems services, with a special emphasis on specification. Our contribution focuses on the development of robust telephony services using DSLs. The collaboration with researchers in formal methods aims to use tools (*e.g.*, theorem provers) to formalize and check properties specific to the DSL and the domain of telephony.

7.2. Ambient Intelligence For The Networked Home Environment (IP6 Amigo)

Participants: Laurent Burgy, Charles Consel, Fabien Latry, Nicolas Palix, Laurent Réveillère.

The Amigo project will focus on the usability of a networked home system by developing open, standardized, interoperable middleware. The developed middleware will guarantee automatic dynamic configuration of the devices and services within this home system by addressing autonomy and composability aspects. The second focus of the Amigo project will be on improving the end-user attractiveness of a networked home system by developing interoperable intelligent user services and application prototypes. The Amigo project will further support interoperability between equipment and services within the networked home environment by using standard technology when possible and by making the basic middleware (components and infrastructure) and intelligent user services available as open source software together with architectural rules for everyone to use.

Our work in the Amigo project is based on our DSL paradigm for protocol-based service families, presented in Section. We aim to develop DSLs for service creation. Indeed, the area of networked home systems, targeted by Amigo, relies on protocols for families of services (*e.g.*, SIP, Session Announcement protocol, and Delivery Multimedia Framework). Furthermore, the underlying software architecture in this area relies on a client-server model. This situation should give us an opportunity to further illustrate our approach to making servers DSL-programmable.

¹A version of this article is to appear in IEEE Percom'08 conference.

7.3. Service Oriented Architecture for Embedded Systems – Industrial Fellowship (CIFRE / Thales)

Participants: Charles Consel, Julien Lancia.

The goal of this project is to design and develop a SOA architecture for embedded systems. More especially, it takes into account 3 levels of adaptation: (1) the component level (contracts on resources, performances...), (2) the coupling of components level (dependence, security...), and (3) the software architecture level (resource management, robustness...). A contract-based component approach will be considered to describe nonfunctional properties, to define mechanisms for coupling of components, and to define control mechanisms when executing elements of a component. This study will be illustrated by a concrete application. The research work should be a step toward solving key problems such as composition of services, security, component adaptation and performance.

7.4. Capability-based DSLs – Région Aquitaine Fellowship

Participants: Laurent Burgy, Charles Consel, Laurent Réveillère.

To answer the fundamental need for innovations in terms of services, existing infrastructures have become increasingly open to external developers. Yet, this openness is done at the expense of the robustness. The aim of this project is to integrate approaches dedicated to finely tuning access to resources into programming languages. This study will introduce a unique DSL to program services whose interface to resources is configured with respect to the different roles of programmers and so their capabilities.

7.5. Designing techniques and tools for developing domain-specific languages – Industrial Fellowship (CIFRE / Thales)

Participants: Charles Consel, Zoé Drey.

The goal of this project is to develop a connection between the domain-specific languages and the model driven engineering. We would like to take profit from methodologies, techniques and tools that come from model driven engineering, in order to ease the design and implementation of a domain-specific language (DSL). In another side, the model driven engineering could be combined with the DSL techniques to complete the pure-model vision in a software engineering process where modelling concepts do not suffice or are not relevant. This work will be illustrated and validated with a concrete case study.

7.6. Language Families for Systems Families (ANR Blanc)

Participants: Charles Consel, Laurent Réveillère.

The goal of our research proposal is to place domain expertise at the centre of the software development process. It is aimed to lift the current limitations of software engineering regarding large scale software production, robustness, reliability, maintenance and evolution of software components. Our key innovation is to introduce a software development process parameterized with respect to a specific domain of expertise. This process covers all the stages of software development and combines the following three emerging approaches:

- Domain-specific modelling, also known as model engineering;
- Domain-specific languages, in contrast with general-purpose languages;
- Generative programming and in particular aspect-oriented programming as a means to transform models and programs.

These three approaches have already demonstrated concrete and well-recognized software engineering benefits, in isolation; their combination will permit to cover the entire software development process dedicated to a specific domain of expertise.

7.7. HomeSIP: development of a SIP-based middleware for home automation (France Telecom)

Participants: Damien Cassou, Charles Consel, Wilfried Jouve, Julien Mercadal, Nicolas Palix.

The goal of this project is to develop a middleware based on an extended version of the SIP protocol to program and deploy home automation applications and to remotely control communicating home services.

In particular, this work aims at creating a platform that brings into play:

- Sensors and actuators connected to the home network,
- A LiveBox or residential gateway which acts as an interface between the home network and the ADSL communication network,
- A software layer, named LiveBox layer, which manages communications between a remote terminal (mobile or fixed) and home services.

8. Other Grants and Activities

8.1. International Collaborations

We have been exchanging visits and publishing articles with the following collaborators.

- Julia Lawall, DIKU, University of Copenhagen (Denmark, Copenhagen).
DSLs, specialization, program analysis.
- Calton Pu, Georgia Institute of Technology (USA, Atlanta).
DSLs and specialization for operating systems.
- Gilles Muller, École des Mines in Nantes (France, Nantes).
DSLs, operating Systems.
- Noha Ibrahim and Frédéric Le Mouë, INSA-Lyon (France, Lyon).
Software architectures for pervasive computing.

8.2. Visits and Invited Researchers

The Phoenix group has been visited by:

- Paul Hudak (Professor in the Department of Computer Science at Yale University, U.S.A.), January 10-3;
- Anne-Françoise Lemeur (Associate Professor in the LIFL laboratory at the University of Lille 1, France), March 5;
- Stéphane Ducasse (Professor in the LISTIC laboratory at the University of Savoie, France), March 12-13;
- Damien Pollet (Postdoctoral Scholar in the LISTIC laboratory at the University of Savoie, France), March 12-13;
- Alexandre Bergel (Postdoctoral Scholar at Trinity College in Dublin, Ireland), March 26;
- Nic Volanschi (Researcher, France), June 11;
- Benjamin Goldberg (Associate Professor in the Computer Science Department at New York University, U.S.A.), June 26;
- Sapan Bhatia (Postdoctoral Scholar at Princeton University, U.S.A.), June 29;
- Anne-Marie Kermarrec (INRIA Senior Researcher, France), September 10;
- Julia Lawall (Associate Professor at the University of Copenhagen (DIKU), Denmark), from November 11 to December 16;

9. Dissemination

9.1. Scientific Community Participation

Charles Consel has been involved in the following events as:

- General Chair of the Sixth International Conference on Generative Programming and Component Engineering (GPCE 2007), October 1-3, 2007 in Salzburg, Austria;
- Program Committee member of the *Sixth International Conference on Aspect-Oriented Software Development* (AOSD 2007), March 14-16, 2007 in downtown Vancouver, British Columbia;
- Program Committee member of *TOOLS EUROPE 2007 - Objects, Models, Components, Patterns*, June 24-27, 2007 in Zurich, Switzerland;
- Program Committee member of the *ACM Conference on Principles, Systems and Applications of IP Telecommunications* (IPTComm'07), July 19-20, 2007 in New-York, U.S.A.;
- Program Committee member of the *22nd International Conference on Object-Oriented Programming, Systems, Languages and Applications* (OOPSLA 2007), October 21-25, 2007 in Montréal, Canada;
- Program Committee member of the *Tenth International Symposium on Practical Aspects of Declarative Languages* (PADL 2008), January 7-8, 2008 in San Francisco, U.S.A., co-located with PoPL 2008;
- Program Committee member of Telecom Track of the *30th International Conference on Software Engineering* (ICSE 2008), May 10-18, 2008 in Leipzig, Germany;
- Committee member of the SPECIF best thesis award (2005-2008);

Charles Consel has participated in the following defense committees (thesis and habilitation) as:

- Committee member for Ali-Erdem Ozcan's PhD thesis, March 28, 2007, University of Grenoble, France;
- Committee member (reviewer) for Jan Midtgaard's PhD thesis, June 20, 2007, University of Aarhus, Denmark;
- Committee member (reviewer) for Mario Sudholt's HDR thesis, July 11, 2007, University of Nantes, France;
- Committee member (reviewer) for Renaud Marlet's HDR thesis, November 23, 2007, University of Bordeaux 1, France.

Laurent Réveillère has been involved in the following events as:

- Program committee member of the *6ème atelier sur les Objets, Composants et Modèles dans l'Ingénierie des Systèmes d'Information*;
- Member of the IFIP working group on *Program Generation*;
- Program committee member of the journal on *Domain-Specific Aspect Languages*;
- Secretary of the French chapter of ACM SIGOPS (ASF);
- External reviewer for PADL'07, GPCE'07 and TOOLS'07.

9.2. Teaching

Charles Consel and Laurent Réveillère have been teaching Master's level courses on:

- Domain-Specific Languages and Program Analysis;
- Telephony over IP (related protocols, the SIP protocol, existing programming interfaces). Students are also offered practical labs on various industrial-strength telephony platforms.

Charles Consel and Laurent Réveillère are also teaching other courses on Operating Systems, Web programming and Compilation.

9.3. Presentations and Invitations

Charles Consel gave a number of invited presentations.

- Invited speaker at Macquarie University, February, 2007, Australia;
- Invited speaker at the 31ème Journée de l'Observatoire Français des Techniques Avancées (OFTA), May 29, 2007, Paris, France;
- Invited speaker at INNOVALIS Aquitaine - TIC et Services aux entreprises - Rencontres Industriels Chercheurs, June 12, 2007, Biarritz, France.

10. Bibliography

Major publications by the team in recent years

- [1] C. CONSEL. *From A Program Family To A Domain-Specific Language*, in "Domain-Specific Program Generation; International Seminar, Dagstuhl Castle", C. LENGAUER, D. BATORY, C. CONSEL, M. ODESKY (editors), Lecture Notes in Computer Science, State-of-the-Art Survey, n° 3016, Springer-Verlag, 2004, p. 19-29, <http://phoenix.labri.fr/publications/papers/dagstuhl-consel.pdf>.
- [2] 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>.
- [3] C. CONSEL, R. MARLET. *Architecting 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, vol. 1490, September 1998, p. 170–194, <http://phoenix.labri.fr/publications/papers/plilp98.ps.gz>.
- [4] 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.
- [5] C. CONSEL, L. RÉVEILLÈRE. *A DSL Paradigm for Domains of Services: A Study of Communication Services*, in "Domain-Specific Program Generation; International Seminar, Dagstuhl Castle", C. LENGAUER, D. BATORY, C. CONSEL, M. ODESKY (editors), Lecture Notes in Computer Science, State-of-the-Art Survey, n° 3016, Springer-Verlag, 2004, p. 165 – 179, http://phoenix.labri.fr/publications/papers/dagstuhl04_consels_reveillere.pdf.
- [6] A.-F. LE MEUR, J. LAWALL, C. CONSEL. *Specialization Scenarios: A Pragmatic Approach to Declaring Program Specialization*, in "Higher-Order and Symbolic Computation", vol. 17, n° 1, 2004, p. 47–92, <http://phoenix.labri.fr/publications/papers/spec-scenarios-hosc2003.ps.gz>.
- [7] 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", vol. 19, n° 2, May 2001, p. 217–251, <http://phoenix.labri.fr/publications/papers/tocs01-namee.pdf>.

- [8] 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>.
- [9] 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, http://phoenix.labri.fr/publications/papers/Reveillere-Muller_dsn2001.pdf.
- [10] S. THIBAUT, C. CONSEL, G. MULLER. *Safe and Efficient Active Network Programming*, in "17th IEEE Symposium on Reliable Distributed Systems, West Lafayette, IN", October 1998, p. 135–143, <http://phoenix.labri.fr/publications/papers/srds98-thibault.ps.gz>.

Year Publications

Articles in refereed journals and book chapters

- [11] S. BHATIA, C. PU, C. CONSEL. *Remote Specialization for Efficient Embedded Operating Systems*, in "ACM Transactions on Programming Languages and Systems", to appear, 2007.

Publications in Conferences and Workshops

- [12] L. BURGY, L. RÉVEILLÈRE, J. LAWALL, G. MULLER. *A Language-Based Approach for Improving the Robustness of Network Application Protocol Implementations*, in "Proceedings of 26th IEEE International Symposium on Reliable Distributed Systems (SRDS'07)", to appear, October 2007.
- [13] C. CONSEL, W. JOUVE, J. LANCIA, N. PALIX. *Ontology-Directed Generation of Frameworks For Pervasive Service Development*, in "Proceedings of the 4th IEEE Workshop on Middleware Support for Pervasive Computing (PerWare'07), White Plains, NY, USA", March 2007, http://phoenix.labri.fr/publications/papers/consel-al_perware-07.pdf.
- [14] C. CONSEL, J. LAWALL, N. PALIX, L. RÉVEILLÈRE. *A Stepwise Approach to Developing Languages for SIP Telephony Service Creation*, in "Proceedings of ACM Conference on Principles, Systems and Applications of IP Telecommunications (IPTComm'07), New-York, NY, USA", July 2007, p. 79–88, http://phoenix.labri.fr/publications/papers/palix-reveillere-al_iptcomm07.pdf.
- [15] W. JOUVE, N. IBRAHIM, L. RÉVEILLÈRE, F. LE MOUËL, C. CONSEL. *Building Home Monitoring Applications: From Design to Implementation into The Amigo Middleware*, in "Proceedings of The Second International Conference on Pervasive Computing and Applications (ICPCA'07), Birmingham, UK", July 2007, <http://phoenix.labri.fr/publications/papers/icpca07.pdf>.
- [16] F. LATRY, J. MERCADAL, C. CONSEL. *Staging Telephony Service Creation: A Language Approach*, in "Proceedings of ACM Conference on Principles, Systems and Applications of IP Telecommunications (IPTComm'07), New-York, NY, USA", July 2007, http://phoenix.labri.fr/publications/papers/latry-al_iptcomm07.pdf.

Internal Reports

- [17] C. CONSEL, W. JOUVE, J. LANCIA, J. LAWALL, N. PALIX. *A Domain-Specific IDL and its Compiler for Pervasive Computing Applications*, Research Report, n^o RR-6213, INRIA, Bordeaux, France, June 2007, <http://hal.inria.fr/inria-00153375>.

References in notes

- [18] CGI: *The Common Gateway Interface*, <http://cgi-spec.golux.com/nca>.
- [19] *Session Initiation Protocol (SIP)*, Request for Comments 3261, March 2001.
- [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.
- [21] S. CHIROKOFF, C. CONSEL, R. MARLET. *Combining Program and Data Specialization*, in "Higher-Order and Symbolic Computation", vol. 12, n^o 4, December 1999, p. 309–335.
- [22] C. CONSEL, F. LATRY, L. RÉVEILLÈRE, P. COINTE. *A Generative Programming Approach to Developing DSL Compilers*, in "Fourth International Conference on Generative Programming and Component Engineering (GPCE), Tallinn, Estonia", R. GLUCK, M. LOWRY (editors), Lecture Notes in Computer Science, vol. 3676, Springer-Verlag, September 2005, p. 29–46.
- [23] 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.
- [24] 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, vol. 1490, September 1998, p. 170–194.
- [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", vol. 17, n^o 1, 2004, 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. THIBAUT, 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. THIBAUT, C. CONSEL. *Efficient Implementations of Software Architectures via Partial Evaluation*, in "Journal of Automated Software Engineering", vol. 6, n^o 4, October 1999, 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", vol. 19, n^o 2, May 2001, p. 217–251.
- [33] 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", September 2000.
- [34] 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.
- [35] 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.
- [36] 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.
- [37] 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, vol. 1628, June 1999, p. 367–390.
- [38] 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.
- [39] U. SCHULTZ, J. LAWALL, C. CONSEL. *Automatic Program Specialization for Java*, in "ACM Transactions on Programming Languages and Systems", vol. 25, n^o 4, 2003, p. 452–499.
- [40] S. THIBAUT, C. CONSEL, J. LAWALL, R. MARLET, G. MULLER. *Static and Dynamic Program Compilation by Interpreter Specialization*, in "Higher-Order and Symbolic Computation", vol. 13, n^o 3, September 2000, p. 161–178.
- [41] S. THIBAUT, C. CONSEL, G. MULLER. *Safe and Efficient Active Network Programming*, in "17th IEEE Symposium on Reliable Distributed Systems, West Lafayette, IN", October 1998, p. 135–143.
- [42] S. THIBAUT, R. MARLET, C. CONSEL. *Domain-Specific Languages: from Design to Implementation – Application to Video Device Drivers Generation*, in "IEEE Transactions on Software Engineering", vol. 25, n^o 3, May 1999, p. 363–377.