INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Team Phoenix

Programming Language Technology For Communication Services

Futurs

THEME COM

Table of contents

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 physically located at LaBRI.

Team Leader
Charles Consel [Professor, ENSEIRB, HdR]

ENSEIRB personnel
Laurent Réveillère [Associate Professor, ENSEIRB]

External collaborator
Julia Lawall [Associate Professor at the University of Copenhagen]

Ph.D. students
Laurent Burgy [From October 1, 2004, regional scholarship]
Fabien Latry [From October 1, 2004, Inria scholarship]
Nicolas Palix [From October 1, 2004, Inria scholarship]
Sapan Bhatia [From January 1, 2003, to June 1, 2006, Inria and regional scholarship]
Wilfried Jouve [From October 3, 2005, Phoenix scholarship]
Julien Lancia [From November 14, 2005, Thales (industrial Ph.D. student)]
Julien Mercadal [From October 2, 2006, ministerial scholarship]
Zoé Drey [From November 2, 2006, Thales (industrial Ph.D. student)]
Yérom-David Bromberg [From December 1, 2006, Inria scholarship]

Former PHD Students
Sapan Bhatia [At the University of Princeton]

Project assistant
Sylvie Embolla [From September 4, 2006, Secretary]

Technical staff
Thomas Rougelot [From September 4, 2006, Associate Engineer]
Antoine Luu [From September 4, 2006, to December 31, 2006, 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.

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 personalizating 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 implementes 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, thttpd, 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. Telephony software - A graphical telephony service creation workshop 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 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.

6. New Results

6.1. Minimizing cache misses in an event-driven network server: A case study of TUX

Participants: Sapan Bhatia, Charles Consel, Julia Lawall.

We analyze the performance of CPU-bound network servers and demonstrate experimentally that the degradation in the performance of these servers under highconcurrency workloads is largely due to inefficient use of the hardware caches. We then describe an approach to speeding up event-driven network servers by optimizing their use of the L2 CPU cache in the context of the TUX web server, known for its robustness to heavy load. Our approach is based on a novel cache-aware memory allocator and a specific scheduling strategy that together ensure that the total working data set of the server stays in the L2 cache. Experiments show that under high concurrency, our optimizations improve the throughput of TUX by up to 40% and the number of requests serviced at the time of failure by 21%. For more information, see: [11].

6.2. Language Technology for Internet-Telephony Service Creation

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

Telephony is evolving at a frantic pace, critically relying on the development of services to offer a host of new functionalities. However, programming Internet telephony services requires an intimate knowledge of a variety of protocols and technologies, which can be a challenge for many programmers. Furthermore, because telephony is a resource heavily relied on, programmability of telephony platforms should not compromise their robustness.

This paper presents an approach to creating telephony services that builds on programming language technology (i.e., language design and implementation, language semantics, and program analysis). We have developed a language, named Session Processing Language (SPL), that offers domain-specific constructs, abstracting over the intricacies of the underlying technologies. By design, SPL guarantees critical properties that cannot be verified in general-purpose languages. SPL relies on a Service Logic Execution Environment for SIP (SIP-SLEE) that introduces a design framework for service development based around the notion of session.

SPL and SIP-SLEE have been implemented and they are now being used to develop and deploy real services, demonstrating the practical benefits of our approach. For more information, see: [12].

6.3. A High-Level, Open Ended Architecture For SIP-based Services

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

Now that Internet Telephony can interact with systems such as databases, e-mail facilities and Web services, it can offer a host of new functionalities. However, developing enriched, real-size services is quite a challenge considering the requirements that must be fulfilled by the service developer. Such developer must (1) have an extensive knowledge on network protocols and distributed systems; (2) be familiar with often large and complex platform APIs (e.g., JAIN); and (3) fully understand the signaling protocol (e.g., SIP) to develop services that do not compromise the processing of the calls, nor the platform.

All these areas of expertise are required by most existing platforms. They offer unrestricted APIs and support mainstream programming languages such as C, C♯and Java. They provide little abstraction, and thus rely on the programmer to manage the intricacies of the underlying technologies (protocols, network layers, and signaling). Other platforms enable service creation through a scripting language, such as CPL and LESS, that offers a restricted expressiveness and mostly targets the creation of individual user services.

We present a high-level architecture of an Application Server for SIP-based services. Our architecture abstracts over the intricacies of the underlying technologies and facilitates both the development and the management of services.

By revolving around an Application Server, our approach allows a uniform and coherent basis of telephony services to be offered to the platform users, regardless of the heterogeneity of their end systems. For more information, see: [13].

6.4. A Multimedia-Specific Approach to WS-Agreement

Participants: Wilfried Jouve, Julien Lancia, Charles Consel.

WS-Agreement offers a general language and protocol to establish agreements between two parties. In principle, this generality enables a wide variety of domains to be covered. Yet, agreement terms need to be developed for each domain, to allow specific requirements to be expressed. When addressing a domain, one can either invent new agreement terms, or leverage on existing approaches.

This paper proposes to extend the WS-Agreement framework to address multimedia content negotiation. This work relies on an existing protocol for multimedia negotiation, widely used in networking and telecommunications, named Session Description Protocol. This protocol is used as a framework to cover a variety of media (audio, video, images...). Besides building on a well-proven technology, our approach naturally allows the interfacing between the Web Services realm and the networkingtelecommunications realm, creating a host of new usage opportunities. Our work is illustrated by two case studies: image/audio downloading and audio/video streaming. For more information, see: [14].

6.5. Efficient Packet Processing in User-Level Operating Systems: A Study of UML

Participants: Sapan Bhatia, Charles Consel.

Network server consolidation has become popular through recent virtualization technology that builds secure, isolated network systems on shared hardware. One of the virtualization techniques used is that of User-level Operating Systems. (ULOSes) However, the isolation and security they bring comes at the price of performance, as virtualization introduces a number of overheads into the system. Such overheads can be surprisingly large, especially for complex OS modules like network protocol stacks. Our studies of the TCP/IP stack in User-mode Linux (UML), an implementation of a ULOS, attribute the resulting slow-downs to three main sources: the execution of privileged code, memory management across layers, and additional instructions to execute. To mitigate these bottlenecks, we present five optimization techniques, improving the network performance significantly, reducing packet processing latency by 60network throughput by three folds. Furthermore, the network throughput of the improved ULOS is comparable to that of native Linux up to gigabit speeds. For more information, see: [15].

6.6. Processing Domain-Specific Modeling Languages: A Case Study in Telephony Services

Participants: Fabien Latry, Julien Mercadal, Charles Consel.

The Domain-Specific Language (DSL) approach is being actively studied from both a software engineering viewpoint and a programming language viewpoint. It is being successfully applied to a variety of areas such as banking, graphics and networking. Yet, the concept of a DSL is still very vague, making both its applicability and implementation dif- ficult.

This paper introduces a layered approach to DSLs where (1) domain experts are provided with Domain-Specific Modeling Languages (DSMLs), requiring no programming skills and (2) implementation experts deal with Domain-Specific Programming Languages (DSPLs) that require a programming background but abstracts over the intricacies of underlying technologies.

By separating domain and implementation concerns, we show that our layered DSL approach enables high-level tools to be used to both compile and reason about DSML programs. Compilation and program verification amount to defining high-level generative processes.

We illustrate our approach with the domain of telephony service creation. We introduce a DSML for service creation and demonstrate the ease of compiling DSML programs using the Stratego/XT program transformation environment. Two compilation processes are defined for DSML programs targeting (1) a DSPL, illustrating a high-level compilation process and (2) the TLA+ specification language, exemplifying the verification of domain-specific properties. For more information, see: [16].

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

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

Filling an environment with a host of devices has been a reality for some time. As well, creating experiments to make users interact such an environment to perform different kinds of tasks is commonly reported in the literature. However, these experiments are based on pervasive computing applications that are tedious to develop because they combine a number of problems ranging from device heterogeneity, to middleware constraints, to lack of programming support. In this paper, we present an approach to integrating the ontological description of a pervasive computing environment into a programming language, namely Java. The entities of a pervasive computing environment are uniformly captured by the notion of services (e.g., devices, software components and applications). Syntactic constructs are provided to developers to define abstract and concrete services. An abstract service defines variations of concrete services (i.e., actual entities). A notion of service inheritance enables abstract services to form an ontological hierarchy. An abstract service specifies semantics properties that characterize variations of concrete services. Furthermore, an abstract service defines the ways in which it can interact with other services. These modes of interaction cover a wide range of situations, including stream-based services. From an 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. We have implemented the ontology-directed framework generator. Frameworks have been generated from an ontological description of pervasive computing environment targeting building management. Examples have been programmed using these frameworks, including resource managers and surveillance services. For more information, see: [17].

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, targetted 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.

7.3. A Platform for the Development of Robust Multimedia Applications in Mobile Terminals – Région Aquitaine

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

The world of mobile communication terminals (MCT), such as telephones, handheld computers and PCs, has witnessed dazzling advances for the last few years. Most of the effort has been focused on improving the hardware capabilities of the devices rather than the applications offering services to the users. However, as wireless technologies (GPRS, UMTS, BlueTooth, WiFI) are increasingly becoming available on these devices, it is critical to offer robust applications that make the best use of the available resources.

This project aims to develop a platform for the development of robust multimedia services on MCT.

7.4. 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.

7.5. 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 ressources 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.

7.6. 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.

7.7. Language Families for Systems Families

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.

7.8. Pending patent

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

The patent is about a "Dispositif d'interconnexion d'un système d'informations d'entreprise(s) à un serveur d'applications d'un système de téléphonie sur IP".

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.

8.2. Visits and Invited Researchers

The Phoenix group has been visited by:

- Peter Thiemann (University of Freiburg, Germany), at the start of April;
- Olivier Danvy (BRICS, University of Aalborg, Denmark), mid-April;
- Ulrik Pagh Schultz (University of Southern, Denmark), at the end of April;
- David Schmidt (Computing and Information Sciences Department, Kansas State University, USA), at the start of June;
- Julia L. Lawall (DIKU, University of Copenhagen, Denmark), from the 19th of November to the 11th of December;

9. Dissemination

9.1. Scientific Community Participation

Charles Consel has been involved in the following events as:

- 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 the *Fifth International Conference on Generative Programming and Component Engineering* (GPCE 2006), October 22-26, 2006 in Portland, Oregon, co-located with OOPSLA 2006;
- Program committee member of *TOOLS EUROPE 2007 - Objects, Models, Components, Patterns*;
- Program Committee of the First International Workshop on Global Integrated Model Management (GaMMa 2006), co-located with the ICSE 2006;
- Program committe of the *2ème Journées sur l'Ingénierie Dirigée par les Modèles* (IDM 2006), June 27-28, 2006, Université des Sciences et Technologies de Lille;
- Association ACM-SIGOPS de France, French chapter of ACM SIGOPS (ASF);
- Committee member of the ASF best thesis award;
- Committee member of the SPECIF best thesis award (2005-2008);
- Committee member for Frederic Jouault's thesis, September 2006, Université de Nantes;
- Committee member for Nicolas Salatge's thesis, December 2006, Institut National Polytechnique de Toulouse (LAAS);
- Committee member for Gustavo Bobeff's thesis, December 2006, Ecole des Mines de Nantes;
- General Chair of the Sixth International Conference on Generative Programming and Component Engineering (GPCE 2007).

Laurent Réveillère has been involved in the following events as:

- Program committee member of the *5ème atelier sur les Objets, Composants et Modèles dans l'Ingénierie des Systèmes d'Information*;
- Member of the IFIP working group on *Program Generation*;
- Secretary of the French chapter of ACM SIGOPS (ASF);
- Committee member of the ASF best thesis award;
- Committee member for David Bromberg's thesis, December 2006, Université de Versailles.

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 Columbia University, June 2006, New York;
- Invited speaker at AT&T Labs Research, June 2006, New Jersey;
- Invited speaker at the Georgia Institute of Technology, July 2006, Atlanta.

Charles Consel was lectures at the *École des Jeunes Chercheurs en Programmation* (EJCP 2006), June 5-9, 2006 in Luchon.

10. Bibliography

Major publications by the team in recent years

[1] C. CONSEL. *Domain-Specific Program Generation; International Seminar, Dagstuhl Castle*, C. LENGAUER, D. BATORY, C. CONSEL, M. ODERSKY (editors). , Lecture Notes in Computer Science, State-of-the-Art Survey, chap. From A Program Family To A Domain-Specific Language, n^{o} 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. *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, 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. *Domain-Specific Program Generation; International Seminar, Dagstuhl Castle*, C. LENGAUER, D. BATORY, C. CONSEL, M. ODERSKY (editors). , Lecture Notes in Computer Science, State-of-the-Art Survey, chap. A DSL Paradigm for Domains of Services: A Study of Communication Services, n^{o} 3016, Springer-Verlag, 2004, p. 165 – 179, http://phoenix.labri.fr/publications/papers/dagstuhl04_consel_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^{o} 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^{o} 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. THIBAULT, 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

Publications in Conferences and Workshops

[11] S. BHATIA, C. CONSEL, J. LAWALL. *Minimizing cache misses in an event-driven network server: A case study of TUX*, in "To appear in Proceedings of the 31st IEEE International Conference on Local Computer Networks (LCN 2006), Tampa, FL", IEEE Computer Society Press, November 2006.

[12] L. BURGY, C. CONSEL, F. LATRY, J. LAWALL, N. PALIX, L. RÉVEILLÈRE. *Language Technology for Internet-Telephony Service Creation*, in "IEEE International Conference on Communications", June 2006, http://phoenix.labri.fr/publications/talks/icc06.pdf.

[13] L. BURGY, C. CONSEL, F. LATRY, N. PALIX, L. RÉVEILLÈRE. *A High-Level, Open Ended Architecture For SIP-based Services*, in "Proceedings of the tenth International Conference on Intelligence in service delivery Networks (ICIN 2006), Bordeaux, France", May 2006, http://phoenix.labri.fr/publications/talks/icin06_pA4.pdf.

[14] W. JOUVE, J. LANCIA, C. CONSEL, C. PU. *A Multimedia-Specific Approach to WS-Agreement*, in "To appear in Proceedings of The 4th IEEE European Conference on Web Services (ECOWS) (ECOWS 2006), Zurich, Switzerland", IEEE Computer Society Press, December 2006, http://phoenix.labri.fr/publications/papers/ecows06_multimedia_ws-agreement.pdf.

[15] Y. KOH, C. PU, S. BHATIA, C. CONSEL. *Efficient Packet Processing in User-Level Operating Systems: A Study of UML*, in "Proceedings of the 31st IEEE Conference on Local Computer Networks (LCN) (LCN 2006), Tampa, FL", IEEE Computer Society Press, November 2006, http://phoenix.labri.fr/publications/papers/koh-pu-bahtia-al_lcn2006.pdf.

[16] F. LATRY, J. MERCADAL, C. CONSEL. *Processing Domain-Specific Modeling Languages: A Case Study in Telephony Services*, in "Proceedings of the 1st GPCE for QoS Provisioning in Distributed Systems (GPCE4QoS), Portland, Oregon USA", October 2006, http://phoenix.labri.fr/publications/talks/latry-algpce4qos06-talks.pdf.

Internal Reports

[17] C. CONSEL, W. JOUVE, J. LANCIA, N. PALIX. *Ontology-Directed Generation of Frameworks For Pervasive Service Development*, Research Report, n^{o} 00111032, INRIA, Bordeaux, France, November 2006, http://hal.inria.fr/inria-00111032.

References in notes

[18] *CGI: The Common Gateway Interface*, http://cgi-spec.golux.com/ncsa.

[19] *Session Initiation Protocol (SIP)*, Request for Comments 2543, 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. 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", 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. THIBAULT, 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. THIBAULT, 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. THIBAULT, 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.