INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Phoenix

Programming Language Technology For Communication Services

Bordeaux - Sud-Ouest

THEME COM

Table of contents

PHOENIX is an INRIA Project-Team joint with University of Bordeaux (UB1 and ENSEIRB) and CNRS (LaBRI, UMR 5800)

1. Team

Faculty Member
Charles Consel [ENSEIRB, Team Leader, Professor (Pr), HdR]
Laurent Réveillère [ENSEIRB, Associate Professor (MCF), till Feb. 2008]

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

Technical Staff
Thomas Rougelot [INRIA, Associate Engineer, till Sep. 2008]
Benjamin Bertran [INRIA, Associate Engineer]
Alexandre Blanquart [INRIA, Expert Engineer]
Fabien Latry [University Bordeaux 1, Research Engineer (IR), till Oct. 2008]

PhD Student
Laurent Burgy [INRIA, CORDI/C, till Apr. 2008]
Nicolas Palix [INRIA, Conseil Régional d'Aquitaine grant, till Aug. 2008]
Wilfried Jouve [INRIA, CORDI/C, since Sep. 2006]
Julien Lancia [Thales, CIFRE, since Nov. 2005]
Julien Mercadal [University Bordeaux 1, MENRT grant, since Oct. 2006]
Zoé Drey [Thales, CIFRE, since Nov. 2006]
Damien Cassou [University Bordeaux 1, MENRT grant, since Oct. 2007]
Julien Bruneau [Thales, CIFRE, since Oct. 2008]
Henner Jakob [INRIA, CORDI/S, since May 2008]

Post-Doctoral Fellow
David Bromberg [INRIA, till Aug. 2008]
Nicolas Loriant [INRIA, till Oct. 2009]

Administrative Assistant
Sylvie Embolla [INRIA]

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 in order to validate our approach.

2.3. Highlights of the year

Siderion Technologies is a spin-off of the Phoenix research group at INRIA. It was 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.

June 11, 2008, Siderion received an award for technology transfer during the second edition of the Innovaday.

July 3, 2008, Wilfried Jouve *et al.* were awarded as the “Best Student Paper” at the IPTComm’08 conference for their article *A SIP-based Programming Framework for Advanced Telephony* [16].

3. Scientific Foundations

3.1. Introduction

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

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

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

3.2. Adaptation Methodologies

Although industry has long recognized the need to develop adaptable programs, methodologies to develop them are still at the research stage. We have presented preliminary results in this area with a detailed study of the applicability of program specialization to various software architectures [30]. Our latest contributions in this area span from a revolutionary approach based on the definition of programming languages, dedicated to a specific problem family, to a direct exploitation of specialization opportunities generated by a conventional programming methodology.

3.2.1. Domain-Specific languages

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

3.2.2. Declaring adaptation

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

3.2.3. Declaring specialization

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

We introduced a declaration language which enables a component developer to express the configurability of a component. The declarations consist of a collection of specialization scenarios that precisely identify what program constructs are of interest for specialization. The scenarios of a component do not clutter the component code; they are defined aside in a *specialization module* [25], [26], [24], [27].

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 [37]. 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 [36].

3.2.5. Specializing software architectures

The sources of inefficiency in software architectures can be identified in the data and control integration of components, because flexibility is present not only at the design level but also in the implementation. We proposed the use of program specialization in software engineering as a systematic way to improve performance and, in some cases, to reduce program size. We studied several representative, flexible mechanisms found in software architectures: selective broadcast, pattern matching, interpreters, layers and generic libraries. We showed how program specialization can be applied systematically to optimize these mechanisms [29], [30].

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) [32]. This approach was based on DSLs and enables rapid development of robust OSes. Such an approach is critically needed in application domains, like appliances, where new products appear at a rapid pace and needs are unpredictable.

3.3.2. Devil - a DSL for device drivers

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

From a software engineering viewpoint, Devil captures domain expertise and systematizes re-use because it offers suitable built-in abstractions [34]. 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 [33]: 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 [35]. 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 [40]. Although covering a very restricted domain, this language was a very successful proof of concept.

3.4. Adaptation Tools and Techniques

To further the applicability of our approach, we have strengthened and extended adaptation tools and techniques. We have produced a detailed description of the key program analysis for imperative specialization, namely binding-time analysis [22]. This analysis is at the heart of our program specializer for C, named Tempo [22]. We have examined the importance of the accuracy of these analyses to successfully specialize existing programs. This study was conducted in the context of systems software [31].

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 [28]. 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 [39]. 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 [36]. 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 [38].

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 evolutions 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. Developing Safe Coordination Services

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

Coordinating entities in a networked environment has always been a significant challenge for software developers. In recent years, however, it has become even more difficult, because devices have increasingly rich capabilities, combining ever larger range of technologies (networking, multimedia, sensors, etc.).

To address this challenge, we are developing a language-based approach for managing the lifecycle of applications coordinating networked entities. Our approach covers the characterization of the networked environment, the specification of coordination applications, the verification of a networked environment and its deployment. It is carried out in practice by a domain-specific language, named Pantaxou.

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 [correspondent], Laurent Réveillère, 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, Charles Consel [correspondent], Julia Lawall.

Event-driven programming has emerged as a standard for implementing 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 the specific structure and behavior of such servers.

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

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

Participants: Charles Consel [correspondent], Laurent Réveillère, Laurent Burgy, Fabien Latry, 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 d'un système d'informations d'entreprise(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, 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: *ADL*, *DSL*, *SIP*, *middleware*, *pervasive*.

Participants: Damien Cassou, Charles Consel, Wilfried Jouve [correspondent], Benjamin Bertran, Julien Bruneau.

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 from evolving and impeding creativity.

5.7.1. DiaSpec – A Domain Specific ADL and its Compiler for Distributed Pervasive Computing Applications

Architecture Description Languages (ADLs) permit the formal declaration of software systems in reusable elements, their externally visible properties, and their relationships. ADLs introduce verifications and tools to supports architecture description and provide portability by remaining disjoint from implementation languages and technologies. As a consequence, most ADLs provide little or no development support.

Moreover, the distributed nature of software systems and the variety of both hardware and software networked entities have raised practical challenges that existing ADLs do not address properly:

- *Abundance of distributed systems technologies*, *e.g.*, RMI, Web Services, Corba. Those entail complex, and specific coding that reduce code reusability.
- *Unpredictable availability of distributed components*, the dynamic (dis)appearance of components and network malfunctions cannot be properly addressed at the implementation level.
- *Unknown runtime configuration* requires encapsulation for components to be arbitrarily distributed.

To address the above issues, we offer a lightweight ADL called DiaSpec and its compiler DiaGen.

- *DiaSpec*, integrates a component-and-connector architecture with Java. DiaSpec allows the declaration of software architecture as a collection of components together with a set of attributes to characterize component instances. A DiaSpec specification also defines the relationships among components along side their declarations. DiaSpec proposes three types of connectors: commands, events and sessions.
- *DiaGen*, the DiaSpec compiler and runtime, performs both static and runtime verifications over DiaSpec declarations and produces a dedicated programming framework that guides and eases the implementation of components. The generated framework is independent of the underlying distributed technology. As of today, DiaGen supports multiple targets: Local, RMI, SIP and a simulation target (the Web Services and the Corba targets being currently in development).

5.7.2. DiaSim – A Parametrized Simulator for Pervasive Computing Applications

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

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

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

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

We are designing and implementing a domain-specific language, called Pantachou, dedicated to the development of coordination services for networked entities. Pantachou has been used to specify a number of coordination scenarios in areas ranging from home automation to telecommunications. The language semantics has been formally defined and a compiler has been developed. The compiler verifies the coherence of a coordination scenario and generates coordination code in Java.

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

Keywords: *DSL*, *orchestration*, *visual programming language*.

Participants: Zoé Drey [correspondent], Julien Mercadal, Charles Consel.

One of the challenges addressed by the area of pervasive computing is to provide tools for end-users or novice programmers, so that they can easily program applications on pervasive environments like home, hospital, museums, or any building that aim at helping their users' everyday life. Pantagruel is a two-step approach that aims at writing high-level specifications of the orchestration of entities in a pervasive computing environment.

Pantagruel aims at easing the description of an orchestration logic between networked entities of a pervasive environment. First, the developer defines a taxonomy of entities that compose the environment, This step provides an abstraction of the entities capabilities and functionalities. Second, the developer defines the orchestration logic in terms of rules. To facilitate its programming, we provide a visual domain-specific language based on the sensor-controller-actuator paradigm. This step is strongly coupled with the first one, guiding the developer in defining rules. Pantagruel brings a high-level layer intended to complement existing tools in the activity of safe orchestration logic description, allowing novice-programmers to prototype pervasive applications. The Pantragruel compiler generates code compliant with the DiaSpec toolset. Pantagruel is being completed by tools aimed at verifying safety properties, like termination, reachability.

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: [14].

6.2. High-level Programming Support for Robust Pervasive Computing Applications

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

We introduce a domain-specific Interface Definition Language (IDL) and its compiler, dedicated to the development of pervasive computing applications. Our IDL 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 programming 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.

For more information, see: [15].

6.3. A SIP-based Programming Framework for Advanced Telephony Applications

Participants: Wilfried Jouve, Nicolas Palix, Charles Consel, Patrice Kadionik.

The scope of telephony is significantly broadening, providing users with a variety of communication modes, including presence status, instant messaging and videoconferencing. Furthermore, telephony is being increasingly combined with a number of non-telephony, heterogeneous resources, consisting of software entities, such as Web services, and hardware entities, such as location-tracking devices. This heterogeneity, compounded with the intricacies of underlying technologies, make the programming of new telephony applications a daunting task.

We propose an approach to supporting the development of advanced telephony applications. To do so, we introduce a declarative language over Java to define the entities of a target telephony application area. This definition is passed to a generator to produce a Java programming framework, dedicated to the application area. The generated frameworks provide service discovery and high-level communication mechanisms. These mechanisms are automatically mapped into SIP, making our approach compatible with existing SIP infrastructures and entities. Our work is implemented and has been validated on various advanced telephony applications.

For more information, see: [16].

6.4. Pantaxou: a Domain-Specific Language for Developing Safe Coordination Services

Participants: Julien Mercadal, Nicolas Palix, Charles Consel, Julia Lawall.

Coordinating entities in a networked environment has always been a significant challenge for software developers. In recent years, however, it has become even more difficult, because devices have increasingly rich capabilities, combining an ever larger range of technologies (networking, multimedia, sensors, etc.).

To address this challenge, we propose a language-based approach to covering the life-cycle of applications coordinating networked entities. Our approach covers the characterization of the networked environment, the specification of coordination applications, the verification of a networked environment and its deployment. It is carried out in practice by a domain-specific language, named Pantaxou.

We present the domain-specific language Pantaxou, dedicated to the development of applications for networked heterogeneous entities. Pantaxou has been used to specify a number of coordination scenarios in areas ranging from home automation to telecommunications. The language semantics has been formally defined and a compiler has been developed. The compiler verifies the coherence of a coordination scenario and generates coordination code in Java.

For more information, see: [17].

7. Contracts and Grants with Industry

7.1. 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.

7.2. 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). On 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.

7.3. Integrating non-functional properties in an Architecture Definition Language and its execution environment – Industrial Fellowship (CIFRE / Thales)

Participants: Charles Consel, Julien Bruneau.

The goal of this project is to add non functional properties in the DiaSpec ADL and in the DiaGen generator. More especially, these non functional properties are considered on three different levels:

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

This work will be illustrated and validated with a concrete application.

7.4. 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.

7.5. 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).
- Gilles Muller, École des Mines in Nantes (France, Nantes).

  DSLs, operating Systems.

8.2. Visits and Invited Researchers

The Phoenix group has been visited by:

- Torbjörn Ekman (Researcher at the Computing Laboratory at University of Oxford, UK), February 28, 2008;
- Henner Jacob (Intern in the Multimedia Communications Lab of the Department of Electrical Engineering and Information Technology of the Technische Universität Darmstadt, Germany), March 3, 2008;
- Nicolas Loriant (Postdoctoral Scholar in ObAsCo Inria research group of the École des Mines de Nantes, France), March 31, 2008;
- Anthony M. Sloane (Associate Professor in the Department of Computing in the Division of Information and Communication Sciences at Macquarie University, Sydney, Australia), September 25, 2008;

9. Dissemination

9.1. Scientific Community Participation

Charles Consel has been involved in the following events as:

- 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 the *Twenty third International Conference on Object-Oriented Programming, Systems, Languages and Applications* (OOPSLA 2008), October 25-29, 2008 in Orlando, U.S.A., speaker at the *DSLs: The Good, the Bad, and the Ugly* panel, author of an article in the conference companion.
- Program Committee member of the *First International Conference on Software Language Engineering* (SLE 2008), Toulouse, France, co-located with MoDELS 2008;
- Program Committee member of the *First International Conference on Model Transformation* (ICMT 2008), July 1-2, 2008 in Zurich, Switzerland, co-located with TOOLS EUROPE and SEAFOOD;
- Program Committee member of the *Second International Conference on Model Transformation* (ICMT 2009);
- Steering Committee chair of the *Seventh ACM International Conference on Generative Programming and Component Engineering* (GPCE 2008), October 19-23, 2008 in Nashville, U.S.A.;
- Reviewer for the “Agence Nationale de la Recherche” (French national agency for the funding of research);
- Chair of the “Commission de recrutement des Ingnieurs Inria” (Recruitment committee of Inria Technical Staff);

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

- Committee member for Pierre-Étienne Moreau HDR thesis, June 13, 2008, Institut National Polytechnique de Lorraine, France;
- Committee member (reviewer) for Fabien Baligand PhD thesis, June 25, 2008, University of Nantes, France;
- Committee member (reviewer) for Carlos Noguera PhD thesis, November 25, 2008, University of Lille, France;

9.2. Teaching

Charles Consel and Laurent Réveillère have been teaching Master 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. These labs are supervised by Benjamin Bertran and Julien Bruneau.

Charles Consel and Damien Cassou are teaching a course on Architecture Description Languages.

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.

- University of Princeton, April 29, 2008, Princeton, New Jersey, U.S.A;
- University of British Columbia, May 1-4 , 2008, Vancouver, Canada;
- University of Utah, July, 2008, Salt Lake City, Utah, U.S.A.;

Damien Cassou gave an invited presentation to the Software Composition Group (SCG) of the Institute of Computer Science and Applied Mathematics (IAM) of the University of Bern, Switzerland, November 26-28, 2008.

9.4. PhD Thesis

Three students of the Phoenix group obtained their PhD in 2008:

- Laurent Burgy, “Approche langage au développement du support protocolaire d’applications réseaux” [11]. Laurent Burgy is currently a post-doc fellow at Princeton (Princeton, New Jersey, U.S.A.).
- Nicolas Palix, “Langages dédiés au développement de services de communications” [13]. Nicolas Palix is currently a post-doc fellow in the Topps group at DIKU (Copenhagen, Danemark).
- Julien Lancia, “Infrastructure orientée service pour le développement d’applications ubiquitaires” [12].

9.5. In the press

Siderion Technologies is a spin-off of the Phoenix research group at INRIA; it builds upon telecommunications technologies developed in the group. The following references are articles and reports in IT-specific and general purpose journals and radios about Siderion:

- VNUnet, June 12, 2007, *Rencontre sur les TIC !* (online article http://www.aquitaineonline.com/actualites-en-aquitaine/sud-ouest/technologies-information-et-communication-en-aquitaine-2007.html);
- Sud Ouest Eco, November 30, 2007, *Aussi simple qu'un coup de fil*;
- VNUnet, January 29, 2008, *Innovation : Siderion personnalise l'accueil téléphonique* (online article http://www.vnunet.fr/news/innovation___siderion_personnalise_l_accueil_telephonique-2026413);
- Bordeaux Iptima, January 31, 2008, *Siderion Technologies propose une solution pour personnaliser l'accueil téléphonique*;
- Le Journal des Télécoms, January 31, 2008, *La création de services téléphoniques à portée de tous*;
- Les échos, February 1, 2008, *Le téléphone qui trouve le bon interlocuteur* (online article http://www.lesechos.fr/info/innovation/4680159.htm);
- BFM – l'atelier numérique, February 2, 2008, *Siderion programme le cheminement des appels téléphoniques* (online article http://www.atelier.fr/telecoms-fai/10/01022008/siderion-route-appels-telephoniques-en-fonction-appelant-35926-.html);
- 01net, February 4, 2008, *Siderion personnalise l'accueil téléphonique des entreprises* (online article http://www.01net.com/editorial/370982/siderion-personnalise-l-accueil-telephonique-des-entreprises/);
- NetEco, February 4, 2008, *Siderion Technologies lance une solution de routage intelligent* (online article http://www.neteco.com/93258-siderion-technologies-solution-routage-intelligent.html);

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. ODERSKY (editors), Lecture Notes in Computer Science, State-of-the-Art Survey, 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. *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. ODERSKY (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_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° 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. 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

Doctoral Dissertations and Habilitation Theses

[11] L. BURGY. *Approche langage au développement du support protocolaire d'applications réseaux*, Ph. D. Thesis, Université de Bordeaux I - LaBRI / INRIA Bordeaux - Sud-Ouest, April 2008.

[12] J. LANCIA. *Infrastructure orientée service pour le développement d'applications ubiquitaires*, Ph. D. Thesis, Université de Bordeaux I - LaBRI / INRIA Bordeaux - Sud-Ouest, December 2008.

[13] N. PALIX. *Langages dédiés au développement de services de communications*, Ph. D. Thesis, Université de Bordeaux I - LaBRI / INRIA Bordeaux - Sud-Ouest, September 2008.

Articles in International Peer-Reviewed Journal

[14] S. BHATIA, C. CONSEL, C. PU. *Remote Specialization for Efficient Embedded Operating Systems*, in "ACM Transactions on Programming Languages and Systems", vol. 30, n° 4, July 2008.

International Peer-Reviewed Conference/Proceedings

[15] W. JOUVE, J. LANCIA, N. PALIX, C. CONSEL, J. LAWALL. *High-level Programming Support for Robust Pervasive Computing Applications*, in "Proceedings of the 6th IEEE Conference on Pervasive Computing

and Communications (PERCOM'08), Hong Kong, China", March 2008, p. 252–255, http://phoenix.labri.fr/publications/talks/jouve-al_percom08_poster.pdf.

[16] W. JOUVE, N. PALIX, C. CONSEL, P. KADIONIK. *A SIP-based Programming Framework for Advanced Telephony Applications*, in "Proceedings of The 2nd Conference on Principles, Systems and Applications of IP Telecommunications (IPTComm'08), LNCS 5310, Heidelberg, Germany", Best Student Paper Award, July 2008, http://phoenix.labri.fr/publications/papers/jouve-al_iptcomm08.pdf.

[17] J. MERCADAL, N. PALIX, C. CONSEL, J. LAWALL. *Pantaxou: a Domain-Specific Language for Developing Safe Coordination Services*, in "Seventh International Conference on Generative Programming and Component Engineering (GPCE), Nashville, Tennessee, USA", October 2008.

References in notes

[18] *CGI: The Common Gateway Interface*, http://www.w3.org/CGI/.

[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, p. 111–119.

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

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

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

[24] 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.

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

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.