INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team COMPOSE

Design and Development of Adaptive Programs and Systems

Futurs

THEME 2A

Table of contents

1. Team

The Compose Project is located in Bordeaux. Compose is a joint project 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 project is physically located at ENSEIRB.

Head of project team

Charles Consel [Professor, ENSEIRB]

Administrative assistants

Simone Dang Van [half-time administrative assistant, ENSEIRB, until September 1, 2003]
Brigitte Larue-Bourdon [half-time administrative assistant, Inria, from September 1, 2003]

ENSEIRB personnel

Anne-Françoise Le Meur [Teaching assistant, ENSEIRB, until August 31, 2003]
Laurent Réveillère [Associate Professor, ENSEIRB]

Inria personnel

Claus Brabrand [Inria Post-doctoral fellow from February 1 to September 30, 2003; then Research scientist (CR), Inria]

Technical staff

Abdelaaziz El Khaoulany [Inria]

Ph.D. students

Sapan Bhatia [From January 1, 2003, Inria and regional scholarship]
Laurence Caillot [From October 1, 2003, Integro Networks (industrial Ph.D. student)]
Hédi Hamdi [Inria and regional scholarship]
Mathieu Minard [Thomson Multimedia (industrial Ph.D. student)]
Luciano Porto Barreto [Graduated June 30, 2003]

2. Overall Objectives

Key words: *partial evaluation*, *specialization*, *program transformation*, *compilation*, *domain-specific languages*, *program analysis*, *program optimization*, *program adaption*, *software architecture*, *software tools*, *operating systems*, *embedded systems*, *telecommunications*, *networking*, *protocols*, *web services*.

2.1.1. Context.

Adaptability can be a key aspect in the design and implementation of a software component, particularly when this component addresses, not a specific problem or need, but a family of problems or needs. This situation raises the usual conflict between genericity and efficiency. On the one hand, genericity is often traded for efficiency in application areas such as operating systems and networking. On the other hand, in software engineering studies, efficiency is commonly overlooked to achieve a high degree of static and dynamic adaptability [25].

Not only does an adaptable system raise challenges for its development, but the process of adapting such a system with respect to a given context also introduces difficulties. The main problems raised by a high degree of adaptability are as follows.

- Usability: parameters may capture many aspects; as a result, they may be complicated and unstructured. This problem becomes worse at the level of a software architecture, where parts of global parameters need to be propagated down to individual software components.

- Conciseness: an increasing number of parameters usually requires the programmer to write repetitive sequences of operations to set up an appropriate context before any invocation. The resulting program may become large and hard to maintain.
- Robustness: complex parameterization with many configuration values, not surprisingly, translates into more error-prone programming.

These problems are well-known limitations of attempts to make systems and components more re-usable [25].

2.1.2. Overview.

The Compose group aims to study new approaches to developing adaptable software components in a specific target area, namely, systems and networking; in particular, multimedia communication services. These approaches are based on methodologies, techniques and tools from software engineering and programming languages. Concretely, we design and implement new languages dedicated to a problem domain to enable the development of concise and robust programs. Also, we develop program analyzes and program optimizations to ensure domain-specific properties and efficiency.

Our research process consists of identifying and analyzing problems in our application area, developing methodologies, techniques and tools to address these problems, and assessing our proposed solutions on real-size cases.

Because of the cross-disciplinary nature of our research, we have had contributions in various communities like systems, networking, software engineering, and programming languages. In practice, the Compose group has produced many prototypes ranging from a compiler for a domain-specific language aimed to specify variations of e-mail services to a program specializer applied to the TCP/IP protocol stack targeted at optimizing system networking programs.

3. Scientific Foundations

Key words: *partial evaluation*, *specialization*, *program transformation*, *adaptability*, *domain-specific languages*, *software engineering*.

Glossary

Partial Evaluation Program transformation that produces a specialized version of a generic program or function, by performing compile-time evaluation of known sub-expressions.

The research done in the Compose group relies on two main approaches:

- Partial evaluation.
- Domain-specific languages.

3.1.1. Partial evaluation.

It is the process that automates program specialization. Specialization is a program transformation that, given some specific execution context, turns a generic program into a specific program, optimizing for speed and/or space. In particular, specializing an interpreter with respect to a given program yields a compiled program where the interpretation layer has been totally removed.

However, program specialization is hard to use by the average programmer. Making it accessible to the average programmer could have a major impact in terms of software engineering. Indeed, instead of hand-optimizing code (which is error-prone and complicates maintenance), the programmer could write adaptable programs (*i.e.*, generic, flexible, configurable...) and produce efficient implementations via program specialization. Adaptable components should improve re-usability and hence software robustness and productivity.

The main approach to program specialization relies on program analysis to determine how a program should be transformed. In particular, this approach consists of performing a dependency analysis, called *binding-time*

analysis [24]. For an imperative language, like the C language studied by Compose, other analyzes are needed to collect various kinds of information such as aliases and side-effects.

3.1.2. Domain-specific languages.

A domain-specific language (DSL) can be viewed as a programming language dedicated to a particular domain or a family of problems [22][30]. It provides appropriate built-in abstractions and notations; it is usually small, more declarative than imperative, and less expressive than a general-purpose language like C or Java. As a result, it may exhibit properties that are crucial for the software industry:

- Productivity: programming, maintenance and evolution are much easier; re-use is systematized.
- Verification: it becomes possible or much easier to automate formal proofs of critical properties of the software: security, safety, real time, etc..

Those features have drawn the attention of rapidly evolving markets (where there is a need for building families of similar software products, *e.g.*, product lines), as well as markets where reactivity or software certification are critical: Internet, cellular phones, smart cards, electronic commerce, embedded systems, bank ATM, etc. Some companies have indeed started to use DSLs in their development process: ATT, Lucent Technologies, Motorola, Philips, etc [30].

Although promising, the DSL approach suffers from the lack of methodology to design and implement these languages. Considering the nature of DSLs, this methodology should combine software engineering and programming language aspects. The Compose group has many years of experience with the design and development of domain-specific languages and has begun addressing DSL methodology issues [22].

4. Application Domains

Key words: *systems*, *networking*, *telecommunications*, *multimedia*, *protocols*.

Adaptability of software systems is a well-identified need in a variety of domains such as networking [29], operating systems [28][26][27], scientific computing [21] and graphics [23].

Applying our adaptation methodologies and tools to real-size cases is a key aspect of our research work. To do so, we focus our effort on a specific application area, namely, systems and networking; in particular on multimedia communication services. Most such software components inherently need to be adaptable because of the family of services they implement. Yet, they have efficiency constraints because they are intensively executed. Multimedia communication services also have strong safety and security requirements; in particular when dealing with third-party service development.

Our goal is to show how our approaches can lead to highly adaptable components without sacrificing neither safety nor efficiency.

The need for adaptation methodologies and tools is confirmed by our industrial partners such as Integro Networks and Thomson Multimedia. They must adapt to ever changing customer needs and rapidly evolving hardware. They cannot sacrifice efficiency, often because of cost constraints.

5. Software

5.1. NOVA—a Platform for Adaptable Multimedia Communication Services

Participants: Charles Consel, Laurent Réveillère [correspondent], Abdelaaziz El Khaoulany, Sapan Bhatia, Claus Brabrand.

Key words: *adaptation*, *mobility*, *communication services*, *multimedia*, *client/server model*.

NOVA is a platform and framework for adaptable multimedia communication services.

The platform is centered around a collection of *programmable servers* [18] for various *service domains*, such as HTTP services, e-mail services, and telephony services.

These servers may be programmed through so-called *service description languages* which are domain-specific languages to offer expressiveness and conciseness without compromising safety and security. Service description programs may thus define service variations adapting the behavior of a service to a particular client. This makes the services sensitive to and capable of adapting to client characteristics like terminal capabilities, network features, user preferences, and evolving needs.

Thus far, we have developed three service description languages; Pems (Section 5.4) for e-mail services, Hades (Section 5.2) for HTTP services, and Call/C (Section 5.3) for telephony services.

NOVA also has an explicit notion of users each with a collection of heterogeneous devices for accessing multimedia services. A user may register and bind service description programs for each of the supported service domains (HTTP, IMAP, VoIP) to the individual devices.

Furthermore, each user has an associated so-called *mobile space* wherein documents may be remotely stored and manipulated to avoid downloading large files and/or unsupported file formats. The documents may be remotely manipulated through an OS-independent file-system interface. Special programmable directories, known as *sinks*, may be instructed to perform certain tasks when files are stored in them; the tasks may range from printing to faxing to format conversion.

The NOVA platform and framework is the ideal context for validating the individual service description languages. However, it has proved a worthy and interesting research topic in its own right.

The NOVA platform implementation is currently an internal prototype, but should be available for download in the near future.

5.2. Hades—A Domain-Specific Language for HTTP Resource Adaptation

Key words: *services*, *HTTP*, *web*, *browsing*, *adaptation*.

Participants: Charles Consel, Claus Brabrand [correspondent].

Hades is a domain-specific language for specifying robust client adaptations of HTTP resources. Hades programs are targeted at degrading HTTP resources, reducing the overall size and detail level of HTML documents and embedded multimedia contents.

Embedded multimedia contents may be downsampled or transcoded in various ways; images may, for instance, be converted to other formats, resized, or turned to black-and-white to save network bandwidth.

Entire HTML fragments may be discarded or *externalized* which means that they are replaced by a hyperlink to a new file containing the externalized fragment. This way, large and complicated documents may be fragmented into smaller pieces, better suited for display on devices with small screens.

A Hades program is often specified relative to a site or a collection of documents with a similar structure. We have a working prototype of the Hades compiler and proxy extension.

5.3. Call/C—A Domain-Specific Language for Robust Internet Telephony Services

Key words: *services*, *telephony*, *adaptation*, *SIP*, *sessions*.

Participants: Charles Consel, Claus Brabrand [correspondent], Laurence Caillot, Laurent Réveillère.

Call/C is a high-level domain-specific language for specifying robust Internet telephony services.

Call/C reconciles programmability and reliability of telephony services, and offers high-level constructs that abstract over intricacies of the underlying protocols and software layers. Call/C 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 Call/C compiler is nearing completion.

5.4. Pems—A Domain-Specific Language for Robust E-Mail Processing

Key words: *services*, *IMAP*, *e-mail*, *adaptation*.

Participants: Charles Consel, Laurent Réveillère [correspondent], Abdelaaziz El Khaoulany.

Pems is a high-level domain-specific language for specifying robust variations of e-mail services based on the Internet Message Access Protocol (IMAP).

Pems programs define *views* at four different hierarchically structured levels; access-point, mailbox, message, and message fields. Individual messages and fields may, depending on various contextual information, be filtered or appropriately transformed. Attached documents may be subjected to format conversion.

We have a working prototype of the Pems compiler and programmable IMAP server.

5.5. Spidle—A Domain-Specific Language for Robust Stream Processing

Key words: *streaming*, *adaptation*, *parallelization*.

Participants: Charles Consel [correspondent], Laurent Réveillère, Hédi Hamdi.

We have designed and implemented a domain-specific language, named Spidle, for specifying steaming applications.

Spidle offers high-level and declarative constructs; compared to general-purpose languages (GPL), it improves robustness by enabling a variety of verifications to be performed.

We have successfully specified a number of standardized and special-purpose streaming applications which are up to 2 times smaller than equivalent programs written in a GPL such as C.

Preliminary results show that compiled Spidle programs are roughly as efficient as the compiled equivalent of C programs. We have a working prototype for the Spidle compiler.

6. New Results

6.1. Automatic Program Specialization of Java

Participant: Charles Consel.

The object-oriented style of programming facilitates program adaptation and enhances program genericness, but at the expense of efficiency. We demonstrate experimentally that state-of-the-art Java compilation technology fails to compensate for the use of object-oriented abstractions to implement generic programs, and that program specialization can be used to eliminate these overheads. We present an automatic program specializer for Java, and demonstrate experimentally that significant speedups in program execution time can be obtained through automatic specialization. Although automatic program specialization could be seen as overlapping with existing optimizer compiler technology, we show that specialization and compiler optimization are in fact complementary. For more information, see: [14].

6.2. Specialization Scenarios: A Pragmatic Approach to Declaring Program Specialization

Participants: Anne-Françoise Le Meur, Charles Consel.

Partial evaluation is a program transformation that automatically specializes a program with respect to invariants. Despite successful application in areas such as graphics, operating systems, and software engineering, partial evaluators have yet to achieve widespread use. One reason is the difficulty of adequately describing specialization opportunities. Indeed, underspecialization or overspecialization often occurs, without any direct feedback as to the source of the problem.

We have developed a high-level, module-based language allowing the program developer to guide the choice of both the code to specialize and the invariants to exploit during the specialization process. To ease the use of partial evaluation, the syntax of this language is similar to the declaration syntax of the target language of the partial evaluator. To provide feedback, declarations are checked during the analyzes performed by

partial evaluation. The language has been successfully used by a variety of users, including students having no previous experience with partial evaluation. For more information, see: [13].

6.3. On the Automatic Evolution of an OS Kernel using Temporal Logic and AOP

Participant: Anne-Françoise Le Meur.

Automating software evolution requires both identifying precisely the affected program points and selecting the appropriate modification at each point. This task is particularly complicated when considering a large program, even when the modifications appear to be systematic. We illustrate this situation in the context of evolving the Linux kernel to support Bossa, an event-based framework for process-scheduler development. To support Bossa, events must be added at points scattered throughout the kernel. In each case, the choice of event depends on properties of one or a sequence of instructions. To describe precisely the choice of event, we propose to guide the event insertion by using a set of rules, amounting to an aspect, that describes the control-flow contexts in which each event should be generated. In this paper, we present our approach and describe the set of rules that allows proper event insertion. These rules use temporal logic to describe sequences of instructions that require events to be inserted. We also give an overview of an implementation that we have developed to automatically perform this evolution. For more information, see: [19].

6.4. From a Program Family to a Domain-Specific Language

Participant: Charles Consel.

An increasing number of domain-specific languages (DSLs) are being developed and successfully used in a variety of areas including networking, telecommunications, and financial products. Yet, the development of a DSL is still an obscure process and its assessment is often partial.

This paper proposes to structure the development of a DSL on the notion of a program family. We outline the main steps of such development. Furthermore, we argue that a program family provides a basis to assess a DSL.

The ideas discussed in this paper are directly based on our experience in developing DSLs for various domains and studying existing ones. We illustrate these ideas with various examples of DSLs. For more information, see: [11].

6.5. A DSL Paradigm for Domains of Services: A Study of Communication Services

Participants: Laurent Réveillère, Charles Consel.

The domain of services for mobile communication terminals has long become a fast-moving target. Indeed, this domain has been affected by a continuous stream of technological advances on aspects ranging from physical infrastructures to mobile terminals. As a result, services for this domain are known to be very unpredictable and volatile. This situation is even worse when considering services relying heavily on multimedia activities (*e.g.,* games, audio and/or video messages, etc.). Such an application area is very sensitive to a large variety of aspects such as terminal capabilities (graphics, CPU, etc.), bandwidth, service provider's billing policies, QoS, and user expectations.

This paper presents a paradigm based on domain-specific languages (DSLs) that enables networking and telecommunication experts to quickly develop robust communication services. Importantly, we propose implementation strategies to enable this paradigm to be supported by existing software infrastructures. For more information, see: [12].

6.6. The Metafront System: Extensible Parsing and Transformation

Participant: Claus Brabrand.

We present the Metafront tool for specifying flexible, safe, and efficient syntactic transformations between languages defined by context-free grammars. The transformations are guaranteed to terminate and to map grammatically legal input to grammatically legal output.

We rely on a novel parser algorithm that is designed to support gradual extensions of a grammar by allowing productions to remain in a natural style and by statically reporting ambiguities and errors in terms of individual productions as they are being added.

Our tool may be used as a parser generator in which the resulting parser automatically supports a flexible, safe, and efficient macro processor, or as an extensible lightweight compiler generator for domain-specific languages. We show substantial examples of both kinds. For more information, see: [16].

6.7. Creating Virtual Soft Devices with User-mode Linux

Participants: Sapan Bhatia, Laurent Réveillère.

Developing device drivers can be highly tedious as it entails direct interaction with hardware devices, which are difficult to analyze in trying to find the cause of unexpected behavior. User-mode Linux simplifies this task by allowing developers to test and debug their device drivers in user-space.

In this paper, we describe a systematic approach to create virtual *soft* devices for the purpose of testing device drivers while they are still in the stage of development. Soft devices run as user-space processes and can have a GUI interface. We have used the existing emulation capabilities of User-mode Linux and extended them by adding some of our own. We have designed a language named *Saint* to specify soft devices, and implemented a virtual coffee-machine soft device as a proof of concept. For more information, see: [15].

6.8. A Programmable Client-Server Model: Robust Extensibility via DSLs

Participants: Charles Consel, Laurent Réveillère.

The client-server model has been successfully used to support a wide variety of families of services in the context of distributed systems. However; its server-centric nature makes it insensitive to fast-changing client characteristics like terminal capabilities, network features, user preferences, and evolving needs.

To overcome this limitation, we present an approach to enable a server to adapt to different clients by making it programmable. A service-description language is used to program server adaptations. This language is designed as a domain-specific language to offer expressiveness and conciseness without compromising safety and security. We show that our approach makes servers adaptable without requiring the deployment of new protocols or server implementations.

We illustrate our approach with the Internet Message Access Protocol (IMAP). An IMAP server is made programmable and a language, named Pems, is introduced to program robust variations of e-mail services.

Our approach is uniformly used to develop a platform for multimedia communication services. This platform is composed of programmable servers for telephony services, e-mail processing, remote-document processing, and stream adapters. For more information, see: [18].

6.9. Spidle: A DSL Approach to Specifying Streaming Applications

Participants: Charles Consel, Hédi Hamdi, Laurent Réveillère.

Multimedia stream processing is a rapidly evolving domain which requires much software development and expects high performance. Developing a streaming application often involves low-level programming, critical memory management, and finely tuned scheduling of processing steps.

To address these problems, we present a domain-specific language (DSL), named Spidle, for specifying steaming applications. Spidle offers high-level and declarative constructs; compared to general-purpose languages (GPL), it improves robustness by enabling a variety of verifications to be performed.

To asses the expressiveness of Spidle in practice, we have used it to specify a number of standardized and special-purpose streaming applications. These specifications are up to 2 times smaller than equivalent programs written in a GPL such as C.

We have implemented a compiler for Spidle. Preliminary results show that compiled Spidle programs are roughly as efficient as the compiled equivalent of C programs. For more information, see: [17]

6.10. Call/C: A Domain-Specific Language for Robust Internet Telephony Services

Participants: Claus Brabrand, Charles Consel.

The convergence of telecommunications and computer networks has added host of new functionalities to telephony services including Web resources, databases, *etc.* Making these rapidly evolving functionalities available to customers critically relies on developing a stream of new telephony services. Fortunately, this convergence has made the programming of telephony services as accessible as the programming of networking services. Yet, an undesirable effect of this new situation is that such a basic commodity as telephony is no longer dependent on thoroughly tested software, developed by certified programmers. Telephony is now exposed to bugs as found in ordinary software development and caused by common deficiencies such as lack of programming experience and insufficient domain expertise.

To reconcile programmability and reliability of telephony, we present a domain-specific language aimed to specify robust services. This language, named Call/C, offers high-level constructs that abstract over intricacies of the underlying protocols and software layers. Call/C 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. For more information, see [20].

7. Contracts and Grants with Industry

7.1. Microsoft Embedded Systems RFP Grant

Participants: Charles Consel, Laurent Réveillère, Claus Brabrand.

The client-server model has been successfully used to support a wide variety of families of services in the context of distributed systems. However, its server-centric nature makes it insensitive to fast changing client characteristics like terminal capabilities, network features, user preferences and evolving needs.

To overcome this key limitation, we present an approach to enabling a server to adapt to different clients by making it programmable. A service-description language is used to program server adaptations. This language is designed as a domain-specific language to offer expressiveness and conciseness without compromising safety and security.

We have implemented an initial prototype based on Linux with programmable servers for telephony services, remote document processing, e-mail message services, and HTTP requests.

In this contract, entitled “NOVA: A Programmable .NET Platform for Multimedia Communication Services”, we propose to port our prototype onto Windows environment and improve it to reach a fully operational NOVA platform as presented in this proposal.

7.2. Microsoft Grant

Participants: Charles Consel, Claus Brabrand.

.NET aims to turn applications into web services. However, the server-centric nature of the model makes it insensitive to fast changing client characteristics like terminal capabilities, network features, user preferences, trends, and evolving needs. As a consequence, .NET services cannot *adapt* to client needs and requirements. From a prospective client, a .NET service can be seen as a set of *opaque* operations that either do the job or are useless.

Our *programmable server* approach [18] intends to widen the scope of applicability of a server by making it programmable and capable of adapting to clients.

.NET and programmable servers are complementary in that once a web service is found using .NET, it could be adapted to client needs and requirements by a service-description program in the context of our approach.

In this contract, entitled "Programmable .NET Services", we propose to validate the notion of programmable .NET services in the context of the Internet Message Access Protocol (IMAP).

7.3. ACI Security COrSS

Participants: Charles Consel, Claus Brabrand, 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 with special emphasis on specification. Relying on refinement and composition, we will study service interaction in the context of telephony services and service derivation by refinement and composition of open systems.

8. Other Grants and Activities

8.1. International Collaborations

We have collaborations with Professor Calton Pu (Georgia Institute of Technology, Atlanta), currently centered around the specialization of operating system components and the design of a domain-specific language, Spidle, for streaming applications.

We have collaboration with DIKU, the University of Copenhagen, Denmark (Julia L. Lawall and Anne-Françoise Le Meur), on various aspects of specialization.

8.2. Visits and Invited Researchers

Julia L. Lawall (DIKU, the University of Copenhagen, Denmark) and James Larus (Microsoft Research, Redmond, WA) have visited the Compose group.

9. Dissemination

9.1. Scientific Community Participation

Charles Consel has been involved in the following events as:

- program commitee member of *ACM SIGPLAN Conference on Programming Languages Design and Implementation* (PLDI 2004);
- program commitee member of *European Conference on Object Oriented Programming* (ECOOP 2003);
- co-organizer of *Dagstuhl School on Domain-Specific Program Generation*, 2003;
- steering committee member of *ACM Sigplan Conference of Generative Programming and Component Engineering* (GPCE 2004); and
- member of the IFIP group on *Domain-Specific Program Generation*, 2004.

9.2. Teaching

Charles Consel and Laurent Réveillère have taught a Master's course on Domain-Specific Languages.

9.3. Presentations and Invitations

Charles Consel has given talks at:

- *ACM Sigplan Conference of Generative Programming and Component Engineering* (GPCE 2003 research talk and invited talk);
- Inria Grenoble;
- Microsoft Research, Seattle;
- Georgia Institute of Technology, Atlanta; and
- *Dagstuhl School on Domain-Specific Program Generation*, 2003

Charles Consel has been a *RNTL referee* 2003, member of the *Inria recruitment jury* 2003, and a member of the *specialist commission* at l'Ecole des Mines de Nantes, 2003.

Claus Brabrand gave a presentation at the *ACM Third Workshop on Language Descriptions, Tools and Applications* (ETAPS/LDTA 2003) and an invited talk with Charles Consel at the *ACM Sigplan Conference of Generative Programming and Component Engineering*.

Laurent Réveillère gave a presentation at the *18th IEEE International Conference on Automated Software Engineering* (ASE 2003) and a talk at the *Dagstuhl School on Domain-Specific Program Generation*, 2003.

Laurent Réveillère received the "Best Ph.D. Thesis Award" from *ACM SIGOPS France*.

10. Bibliography

Major publications by the team in recent years

[1] C. CONSEL. *Domain-Specific Program Generation; Proceedings of Dagstuhl School.* Springer-Verlag, 2003, chapter From A Program Family To A Domain-Specific Language, to appear.

[2] C. CONSEL, L. HORNOF, J. LAWALL, R. MARLET, G. MULLER, J. NOYÉ, S. THIBAULT, N. VOLANSCHI. *Tempo: Specializing Systems Applications and Beyond.* in « ACM Computing Surveys, Symposium on Partial Evaluation », number 3, volume 30, 1998.

[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 », series Lecture Notes in Computer Science, number 1490, C. PALAMIDESSI, H. GLASER, K. MEINKE, editors, pages 170-194, Pisa, Italy, September, 1998, Article invité.

[4] C. CONSEL, F. NOËL. *A General Approach for Run-Time Specialization and its Application to C.* in « Conference Record of the 23rd Annual ACM SIGPLAN-SIGACT Symposium on Principles Of Programming Languages », ACM Press, pages 145-156, St. Petersburg Beach, FL, USA, January, 1996.

[5] C. CONSEL, L. RÉVEILLÈRE. *A Programmable Client-Server Model: Robust Extensibility via DSLs.* in « Proceedings of the 18th IEEE International Conference on Automated Software Engineering (ASE 2003) », IEEE Computer Society Press, pages 70–79, Montréal, Canada, November, 2003.

[6] R. MARLET, S. THIBAULT, C. CONSEL. *Efficient Implementations of Software Architectures via Partial Evaluation.* in « Journal of Automated Software Engineering », number 4, volume 6, October, 1999, pages 411-440.

[7] G. MULLER, R. MARLET, E. VOLANSCHI, C. CONSEL, C. PU, A. GOEL. *Fast, Optimized Sun RPC Using Automatic Program Specialization.* in « Proceedings of the 18th International Conference on Distributed Computing Systems », IEEE Computer Society Press, pages 240-249, Amsterdam, The Netherlands, May, 1998.

[8] 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) », USENIX Association, pages 17-30, October, 2000.

[9] C. PU, T. AUTREY, A. BLACK, C. CONSEL, C. COWAN, J. INOUYE, L. KETHANA, J. WALPOLE, K. ZHANG. *Optimistic Incremental Specialization: Streamlining a Commercial Operating System.* in « Proceedings of the 1995 ACM Symposium on Operating Systems Principles », ACM Operating Systems Reviews, 29(5), ACM Press, pages 314-324, Copper Mountain Resort, CO, USA, December, 1995.

[10] S. THIBAULT, C. CONSEL, G. MULLER. *Safe and Efficient Active Network Programming.* in « 17th IEEE Symposium on Reliable Distributed Systems », pages 135-143, West Lafayette, Indiana, October, 1998.

Articles in referred journals and book chapters

[11] C. CONSEL. *Domain-Specific Program Generation; Proceedings of Dagstuhl School.* Springer-Verlag, 2003, chapter From A Program Family To A Domain-Specific Language, to appearwith revisions.

[12] C. CONSEL, L. RÉVEILLÈRE. *Domain-Specific Program Generation; Proceedings of Dagstuhl School.* Springer-Verlag, 2003, chapter A DSL Paradigm for Domains of Services: A Study of Communication Services, to appearwith revisions.

[13] A.-F. LE MEUR, J. LAWALL, C. CONSEL. *Specialization Scenarios: A Pragmatic Approach to Declaring Program Specialization.* in « Higher-Order and Symbolic Computation, », 2003, to appear.

[14] U. SCHULTZ, J. LAWALL, C. CONSEL. *Automatic Program Specialization for Java.* in « ACM Transactions on Programming Languages and Systems », number 4, volume 25, 2003, pages 452–499.

Publications in Conferences and Workshops

[15] S. BHATIA, L. RÉVEILLÈRE. *Creating Virtual Soft Devices with User-mode Linux.* in « Sixth NordU/USENIX Conference », Copenhagen, Denmark, 2003, to appear.

[16] C. BRABRAND, M. SCHWARTZBACH, M. VANGGAARD. *The metafront System: Extensible Parsing and Transformation.* in « Proceedings of the Third Workshop on Language Descriptions, Tools and Applications (LDTA 2003) », Warsaw, Poland, April, 2003.

[17] C. CONSEL, H. HAMDI, L. RÉVEILLÈRE, L. SINGARAVELU, H. YU, C. PU. *Spidle: A DSL Approach to Specifying Streaming Application.* in « Second International Conference on Generative Programming and

Component Engineering », Erfurt, Germany, September, 2003.

[18] 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) », IEEE Computer Society Press, pages 70–79, Montréal, Canada, November, 2003.

[19] R. A. ÅBERG, J. L. LAWALL, M. SÜDHOLT, G. MULLER, A.-F. LE MEUR. *On the automatic evolution of an OS kernel using temporal logic and AOP.* in « Proceedings of the 18th IEEE International Conference on Automated Software Engineering (ASE 2003) », IEEE Computer Society Press, pages 196–204, Montreal, Canada, October, 2003.

Internal Reports

[20] C. BRABRAND, C. CONSEL. *Call/C: A Domain-Specific Language for Robust Internet Telephony Services.* Technical report, number 1275-03, INRIA/LaBRI, University of Bordeaux I, October, 2003.

Bibliography in notes

[21] A. BERLIN. *Partial Evaluation Applied to Numerical Computation.* in « Proceedings of the 1990 ACM Conference on LISP and functional programming », pages 139–150, 1990.

[22] 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 », series lncs, volume 1490, C. PALAMIDESSI, H. GLASER, K. MEINKE, editors, pages 170–194, Pisa, Italy, September, 1998.

[23] B. GUENTER, T. KNOBLOCK, E. RUF. *Specializing Shaders.* in « Computer Graphics Proceedings », series Annual Conference Series, ACM Press, pages 343–350, 1995.

[24] N. JONES, C. GOMARD, P. SESTOFT. *Partial Evaluation and Automatic Program Generation.* series International Series in Computer Science, Prentice-Hall, June, 1993.

[25] R. MARLET, S. THIBAULT, C. CONSEL. *Efficient Implementations of Software Architectures via Partial Evaluation.* in « Journal of Automated Software Engineering », number 4, volume 6, October, 1999, pages 411–440.

[26] G. MULLER, R. MARLET, E. VOLANSCHI, C. CONSEL, C. PU, A. GOEL. *Fast, Optimized Sun RPC Using Automatic Program Specialization.* in « Proceedings of the 18th International Conference on Distributed Computing Systems », IEEE Computer Society Press, Amsterdam, The Netherlands, May, 1998.

[27] 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 », pages 17–30, San Diego, California, October, 2000.

[28] C. PU, T. AUTREY, A. BLACK, C. CONSEL, C. COWAN, J. INOUYE, L. KETHANA, J. WALPOLE, K. ZHANG. *Optimistic Incremental Specialization: Streamlining a Commercial Operating System.* in « Proceedings of the 1995 ACM Symposium on Operating Systems Principles », pages 314–324, Copper

Mountain Resort, CO, USA, December, 1995.

[29] S. THIBAULT, C. CONSEL, G. MULLER. *Safe and Efficient Active Network Programming.* in « 17th IEEE Symposium on Reliable Distributed Systems », pages 135–143, West Lafayette, Indiana, October, 1998.

[30] A. VAN DEURSEN, P. KLINT, J. VISSER. *Domain-Specific Languages: An Annotated Bibliography.* in « ACM SIGPLAN Notices », number 6, volume 35, June, 2000, pages 26–36.