IN PARTNERSHIP WITH:
CNRS

Université de Bordeaux

Activity Report 2012

Project-Team PHOENIX

Programming Language Technology For Communication Services

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

RESEARCH CENTER
Bordeaux - Sud-Ouest

THEME
Distributed Systems and Services

Table of contents

Project-Team PHOENIX

Keywords: Programming Languages, Object Oriented Programming, Verification, Software Engineering, Pervasive Computing

Creation of the Project-Team: September 08, 2005 .

1. Members

Research Scientist
Emilie Balland [Junior Researcher, Inria]

Faculty Members
Charles Consel [Team Leader, Professor, ENSEIRB, HdR]
Hélène Sauzéon [Associate Professor, on leave from University Bordeaux Segalen, from September 2012]
Bernard N'Kaoua [Professor, University Bordeaux Segalen (Associate Member)]

External Collaborator
Julia Lawall [Senior Researcher, REGAL project-team]

Engineers
Amélie Marzin [Associate Engineer, from October 3, 2011 to July 19, 2012]
Julien Bruneau [Expert Engineer, from March 5, 2012]
Damien Martin-Guillerez [Expert Engineer, from March 1, 2012]

PhD Students
Hongyu Guan [Region scholarship, from February 9, 2009, to February 8, 2012]
Pengfei Liu [Inria scholarship, from October 1, 2009, to September 30, 2012]
Stéphanie Gatti [Thales scholarship, from February 1, 2010]
Quentin Enard [Thales scholarship, from February 1, 2010]
Luc Vercellin [Inria scholarship, from October 3, 2011, to October 2, 2012]

Post-Doctoral Fellows
Young-Joo Moon [Inria scholarship, from November 2, 2011 to November 1, 2012]
Christine Louberry [scholarship funded by the "Biomasud" project, from September 1, 2011 to June 30, 2012]

Administrative Assistant
Chrystel Plumejeau [Group Assistant, from October 3, 2011]

2. Overall Objectives

2.1. Overall Objectives

The Phoenix research group and its predecessor, Compose, have pioneered principles and techniques in *Domain-Specific Languages* (DSLs) [33], [28], and validated the DSL approach in a wide range of application areas, including operating systems [41], [45], [46], networking [29], [49], telecommunications [38], [43], and multimedia streaming [27], [50]. Although successful, the DSL approach is still mostly confined to specific application areas; its dissemination is often impeded by a syntactic and semantic gap with mainstream programming languages; and, most DSLs suffer from much rudimentary programming support and development environments, compared to their industrial-strength equivalent for mainstream programming languages.

To pursue further our research in DSLs, in 2008, we set out to combine the benefits of the DSL approach with the rich support of mainstream programming languages. Combining the DSL approach with a mainstream programming language is aimed to bring high-level, domain-specific concepts into programming, while leveraging high-quality compilers, opulent integrated-development environments, and massive APIs. To maximally leverage the existing language support, this combination should not impact the mainstream language, and thus preclude language embedding.

Specifically, our overall goal is to introduce a design-driven approach to software development, allowing a software system to be designed prior to being programmed. Our approach is meant to be programming-language inspired, defining syntax and semantics for the design language, and leveraging a design artifact to verify conformance properties and to generate programming support.

2.2. Highlights of the Year

- Our first user experiments in the domain of digital assistance:
  - Experimental evaluation of a digital assistance for school inclusion of autistic children (first deployment in the *Gérard Philipe* College in Pessac from September 2012),
  - Need analysis and pre-evaluation of DiaSuiteBox with 80 elderly persons, in collaboration with the UDCCAS Gironde (Union Départementale des Centres Communaux d'Action Sociale) managing elderly care and the "Université du Temps Libre" in Bordeaux,
  - Experimental evaluation of a cognitive assistance for supporting the autonomy of persons with intellectual disabilities, in collaboration with the TSA Chair of UQTR (Université du Québec à Trois-Rivières).

  These experiments are supervised by Hélène Sauzéon, a researcher in Cognitive Science member of the PHOENIX project-team, on leave from the University of Bordeaux 2 since September 2012.
- The DiaSuiteBox project has been accepted to the startup accelerator program "Le Camping" in Toulouse. This program allows 6 startup projects to be mentored by experienced entrepreneurs during 6 months.
- Five articles accepted in top-ranked journals (IEEE Transactions on Software Engineering, Visual Languages and Computing, 2 Software Practice and Experience, and Science of Computer Programming).

3. Scientific Foundations

3.1. Design-Driven Software Development

Raising the level of abstraction beyond programming is a very active research topic involving a range of areas, including software engineering, programming languages and formal verification. The challenge is to allow design dimensions of a software system, both functional and non-functional, to be expressed in a high-level way, instead of being encoded with a programming language. Such design dimensions can then be leveraged to verify conformance properties and to generate programming support.

Our research on this topic is to take up this challenge with an approach inspired by programming languages, introducing a full-fledged language for designing software systems and processing design descriptions both for verification and code generation purposes. Our approach is also DSL-inspired in that it defines a conceptual framework to guide software development. Lastly, to make our approach practical to software developers, we introduce a methodology and a suite of tools covering the development life-cycle.

To raise the level of abstraction beyond programming, the key approaches are model-driven engineering and architecture description languages. A number of *architecture description languages* have been proposed; they are either (1) coupled with a programming language (*e.g.,* [48]), providing some level of abstraction above programming, or (2) integrated into a programming language (*e.g.,* [20], [51]), mixing levels of abstraction. Furthermore, these approaches poorly leverage architecture descriptions to support programming, they are crudely integrated into existing development environments, or they are solely used for verification purposes. *Model-driven software development* is another actively researched area. This approach often lacks code generation and verification support. Finally, most (if not all) approaches related to our research goal are *general purpose*; their universal nature provides little, if any, guidance to design a software system. This situation is a major impediment to both reasoning about a design artifact and generating programming support.

3.2. Integrating Non-Functional Concerns into the Design of Software Systems

Most existing design approaches do not address non-functional concerns. When they do, they do not provide an approach to non-functional concerns that covers the entire development life-cycle. Furthermore, they usually are general purpose, impeding the use of non-functional declarations for verification and code generation. For example, the Architecture Analysis & Design Language (AADL) is a standard dedicated to real-time embedded systems [31]. AADL provides language constructs for the specification of software systems (*e.g.,* component, port) and their deployment on execution platforms (*e.g.,* thread, process, memory). Using AADL, designers specify non-functional aspects by adding properties on language constructs (*e.g.,* the period of a thread) or using language extensions such as the Error Model Annex. [1] The software design concepts of AADL are still rather general purpose and give little guidance to the designer.

Beyond offering a conceptual framework, our language-based approach provides an ideal setting to address non-functional properties (*e.g.,* performance, reliability, security, ...). Specifically, a design language can be enriched with non-functional declarations to pursue two goals: (1) expanding further the type of conformance that can be checked between the design of a software system and its implementation, and (2) enabling additional programming support and guidance.

We are investigating this idea by extending our design language with non-functional declarations. For example, we have addressed error handling [9], access conflicts to resources [34], and quality of service constraints [32].

Following our approach to paradigm-oriented software development, non-functional declarations are verified at design time, they generate support that guides and constrains programming, they produce a runtime system that preserves invariants.

4. Application Domains

4.1. Introduction

Building on our previous work, we are studying software development in the context of communication services, in their most general forms. That is, going beyond human-to-human interactions, and covering human-to-machine and machine-to-machine interactions. Software systems revolving around such forms of communications can be found in a number of areas, including telephony, pervasive computing, and assisted living; we view these software systems as coordinating the communication between networked entities, regardless of their nature, human, hardware or software. In this context, our three main application domains are pervasive computing, avionics and assisted living.

[1] The Error Model Annex is a standardized AADL extension for the description of errors [52].

4.2. Pervasive Computing

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

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

4.3. Assisted Living

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

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

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

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

4.4. Avionics

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

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

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

5. Software

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

Participants: Charles Consel [correspondent], Julien Bruneau, Amélie Marzin, Damien Martin-Guillerez, Emilie Balland.

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

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

- *Defining an application area.* First, an expert defines a catalog of entities, whether hardware or software, that are specific to a target area. These entities serve as building blocks to develop applications in this area. They are gathered in a taxonomy definition, written in the taxonomy layer of the DIASPEC language.
- *Designing an application.* Given a taxonomy, the architect can design and structure applications. To do so, the DIASPEC language provides an application design layer [40]. This layer is dedicated to an architectural pattern commonly used in the pervasive computing domain [30]. Describing the architecture application allows to further model a pervasive computing system, making explicit its functional decomposition.
- *Implementing an application.* We leverage the taxonomy definition and the architecture description to provide dedicated support to both the entity and the application developers. This support takes the form of a Java programming framework, generated by the DIAGEN compiler. The generated programming framework precisely guides the developer with respect to the taxonomy definition and the architecture description. It consists of high-level operations to discover entities and interact with both entities and application components. In doing so, it abstracts away from the underlying distributed technologies, providing further separation of concerns.
- *Testing an application.* DIAGEN generates a simulation support to test pervasive computing applications before their actual deployment. An application is simulated in the DIASIM tool, without requiring any code modification. DIASIM provides an editor to define simulation scenarios and a 2D-renderer to monitor the simulated application. Furthermore, simulated and actual entities can be mixed. This hybrid simulation enables an application to migrate incrementally to an actual environment.
- *Deploying a system.* Finally, the system administrator deploys the pervasive computing system. To this end, a distributed systems technology is selected. We have developed a back-end that currently targets the following technologies: Web Services, RMI, SIP and OSGI. This targeting is transparent for the application code. The variety of these target technologies demonstrates that our development approach separates concerns into well-defined layers.

This development cycle is summarized in the Figure 1.

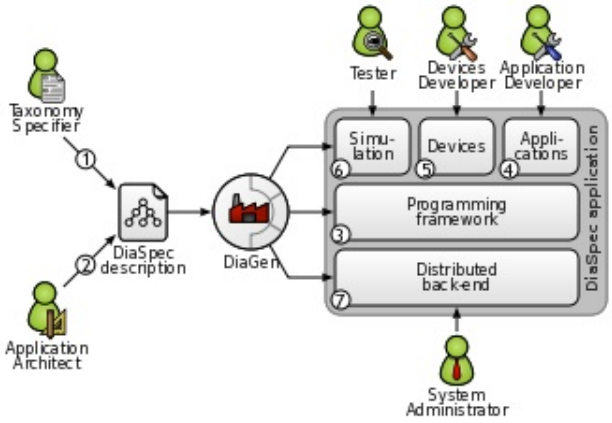

Figure 1. DIASUITE *Development Cycle*

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

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

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

- DIASPEC is composed of two layers:
  - The *Taxonomy Layer* allows the declaration of entities that are relevant to the target application area. An entity consists of sensing capabilities, producing data, and actuating capabilities, providing actions. Accordingly, an entity description declares a data source for each one of its sensing capabilities. As well, an actuating capability corresponds to a set of method declarations. An entity declaration also includes attributes, characterizing properties of entity instances. Entity declarations are organized hierarchically allowing entity classes to inherit attributes, sources and actions. A taxonomy allows separation of concerns in that the expert can focus on the concerns of cataloging area-specific entities. The entity developer is concerned about mapping a taxonomical description into an actual entity, and the application developer concentrates on the application logic.
  - The *Architecture Layer* is based on an architectural pattern commonly used in the pervasive computing domain [30]. It consists of context components fueled by sensing entities. These components process gathered data to make them amenable to the application needs. Context data are then passed to controller components that trigger actions on entities. Using an architecture description enables the key components of an application to be identified, allowing their implementation to evolve with the requirements (*e.g.,* varying light management implementations in a controller component to optimize energy consumption).
- DIAGEN is the DIASPEC compiler that performs both static and runtime verifications over DIASPEC declarations and produces a dedicated programming framework that guides and eases the implementation of components. The generated framework is independent of the underlying distributed technology. As of today, DIAGEN supports multiple targets: Local, RMI, SIP, Web Services and OSGI.

5.1.2. DiaSim: a Parametrized Simulator for Pervasive Computing Applications

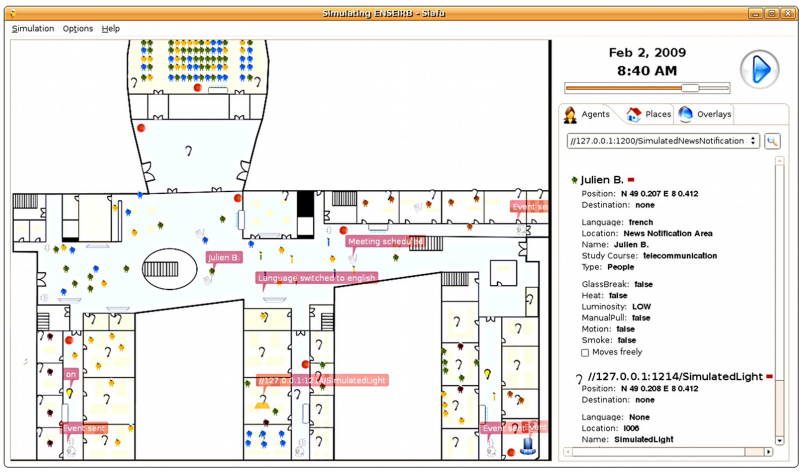

Figure 2. A screenshot of the DIASIM *simulator*

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

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

5.2. DiaSuiteBox: an Open Service Platform

Participants: Julien Bruneau [correspondent], Damien Martin-Guillerez, Charles Consel, Emilie Balland.

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

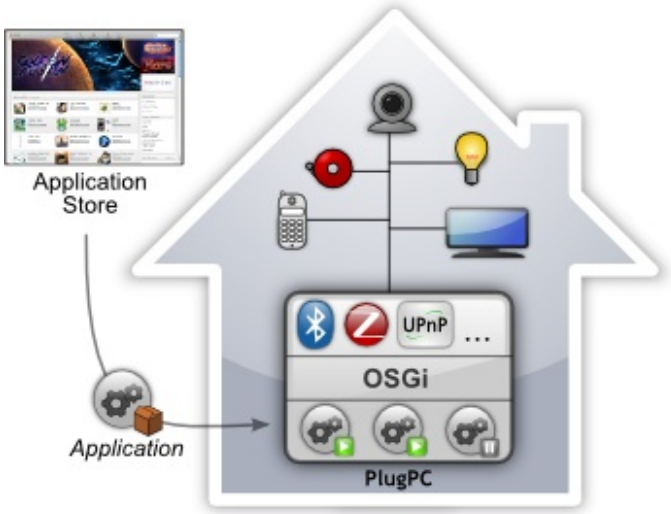

Figure 3. DiaSuiteBox platform architecture

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

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

6. New Results

6.1. Design-driven Testing by simulation

Previously, we have introduced a paradigm-oriented development approach that revolves around a conceptual framework concretized by a design language [26]. A design description is used to generate high-level programming support, to perform a range of verifications, and to abstract over underlying technologies.

This approach is illustrated with the Sense-Compute-Control (SCC) paradigm [48], where an SCC software system gathers information about an environment via sensors (whether hardware or software) and issues orders to impact the environment via actuators. The SCC paradigm has a wide spectrum of applicability; we have used it successfully in the domains of home/building automation, multimedia, avionics and networking.

SCC systems involve both software concerns, like any software system, and integration concerns, for the constituent networked entities forming the environment of the SCC-loop. This situation is problematic for testing 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.

We have developed a simulation approach and a tool named *DiaSim* that leverage the DiaSpec description of an environment [15]. This description is used to generate both a programming framework to develop the simulation logic and an emulation layer to execute applications. The generic nature of our approach has been illustrated by leveraging two different simulation tools, namely, Siafu for 2-D rendering of home/building spaces, and FlightGear for avionics.

To fuel the simulation of an environment with accurate stimuli, we need to model real systems, including natural phenomena (*e.g.,* heat transfer in a building) or mechanical systems (*e.g.,* aircraft models). These physical models are typically defined as continuous systems using differential equations. To facilitate the reuse of off-the-shelf physical models, we have used a DSL named Acumen for describing differential equations. Acumen continuous models are coupled with the DiaSim discrete simulator, forming a hybrid system fueled by accurate stimulus producers [18].

These major accomplishments were conducted by Julien Bruneau, in the context of his PhD studies [11].

6.2. Design-driven Development of Dependable Software Systems

Dependability of a system is the ability to avoid service failures that are more frequent and more severe than is acceptable [22]. This generic concept includes attributes such as availability, integrity and reliability. Dependable systems are now pervasive in a range of domains (*e.g.,* railway, avionics, automotive) and require a certification process. The main goal of certification is to demonstrate that a system is conform to its *high-level requirements*, resulting from functional and safety analyses.

Software plays an increasingly important role in dependable systems; software development is thus required to be certified. In particular, the stakeholders have to pay attention to the coherence of the functional and non-functional aspects of an application to demonstrate the conformance of the software with the high-level requirements. Non-functional aspects of a system refer to constraints on the manner in which this system implements and delivers its functionality (*e.g.,* performance, reliability, security) [48].
Coherence. Because functional and non-functional aspects are inherently coupled, ensuring their coherence is critical to avoid unpredicted failures [39]. For example, fault-tolerance mechanisms may significantly deteriorate the application performance. Generally, this kind of issues are detected at the late stages of the development process, increasing the development cost of applications [21].
Conformance. Ensuring that an application is in conformance with its high-level requirements is typically done by tracing their propagation across the development stages. In practice, this process is human-intensive and error prone because it is performed manually [37].

Certifying a development process requires a variety of activities. In industry, the usual procedures involve holding peer review sessions for coherence verification, and writing traceability documents for conformance certification. In this context, *design-driven development* approaches are of paramount importance because the design drives the development of the application and provides a basis for tracing requirements [53]. However, because most existing approaches are general purpose, their guidance is limited, causing inconsistencies to be introduced in the design and along the development process. This situation calls for an integrated development process centered around a conceptual framework that allows to guide the certification process in a systematic manner. In response to this situation, we proposed a design-driven development methodology, named DIASUITE [2], which is dedicated to the *Sense/Compute/Control (SCC) paradigm* [48]. As demonstrated by Shaw, the use of a specific paradigm provides a conceptual framework, leading to a more disciplined engineering process and guiding the verification process [47]. An SCC application is one that interacts with a physical environment. Such applications are typical of domains such as home/building automation, robotics and avionics.

In this work, we have shown the benefits of DIASUITE for the development of dependable SCC applications. This approach is applied to a realistic case study in the avionics domain, in the context of two non-functional aspects, namely time-related performance and reliability. The DIASUITE design language, named DIASPEC , offers declarations covering both functional and non-functional dimensions of an SCC application [2], [9] [32]. However, so far, the DIASUITE methodology has only been used to study each dimension in isolation, leaving open the problems of coherence and conformance when considering multiple dimensions. This work integrates all these dimensions, enabling the generation of validation support. More precisely, this work makes the following contributions:
Design coherence over functional and non-functional dimensions. We use the DIASPEC language to describe both functional and non-functional aspects of an application and apply this approach to a realistic case study. A DIASPEC description is verified at design time for coherence of its declarations. This verification is performed with respect to a formal model generated from a DIASPEC description.
Design conformance through the development process. At design time, we provide verification support to check the conformance between the specification and the formalized form of the high-level requirements. At implementation time, we guarantee the conformance between the application code and the previously verified requirements. This process is automatically done by leveraging the generative approach of DIASUITE . As some of the high-level requirements cannot be ensured at design time (*e.g.,* time-related performance), we provide further testing support to validate the implementation with respect to these remaining requirements. This support leverages a realistic flight simulator, namely FlightGear [44].
Validation in avionics. We validate our approach by developing a realistic case study in avionics. Following the DIASUITE methodology, we have developed an aircraft flight guidance system and tested it on FlightGear. Additionally, we have duplicated this case study in the context of a commercial drone system, namely Parrot AR.Drone. [2]

These accomplishments were conducted by Julien Bruneau, Quentin Enard and Stéphanie Gatti, in the context of their PhD studies. This work will be published at the International Conference on Pervasive and Embedded Computing and Computation Systems (PECCS'13).

6.3. Putting DiaSuite to Work

A continuing concern of the Phoenix research group is to put our work into practice by tackling realistic applications. We have validated DiaSuite on a variety of applications in areas including telecommunications, pervasive computing, and avionics.

Our expertise in smart home and building, combined with the maturity of DiaSuite, have given rise to the development of a dedicated instance of our technology called DiaSuiteBox. This instance is destined for technology transfer.

[2] http://ardrone.parrot.com

6.3.1. Applying DiaSuite to a Variety of Areas.

Let us examine the application of DiaSuite to two key areas: pervasive computing and avionics. In each case, demonstrations and posters have been presented to researchers and industrial partners [24], [25], [23], [35]

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

At the beginning of this evaluation period, our research group was mainly interested in orchestrating applications in the telecommunications domain, leveraging new opportunities created by the emergence of Voice over IP (mainly based on SIP). Concurrently, a myriad of objects became networked, prompting a need to expand the scope of telecommunications beyond human-human interaction.

Two main industrial collaborations were instrumental to explore the scope of this evolution and to validate the Diasuite approach with realistic case studies. First, we collaborated with a French telecommunications company, in a two-year project named HomeSIP, to study the convergence between VoIP and networked objects in the context of home automation. During this project, we developed a range of applications, including remote appliance control through phone keypad, TV recording via SMS, and dynamic entry phone systems. Second, we contributed to a two-year project named SmartImmo, which gathered major French companies in the area of building construction, installation, and management. The goal of this project was to create a service infrastructure for building automation. SmartImmo gave us the opportunity to elaborate realistic building automation scenarios (*e.g.,* parking lot management, meeting room reservation, energy monitoring).

Our work on applying DiaSuite to the pervasive computing domain has leveraged key contributions by two PhD students of Phoenix, namely, Wilfried Jouve [36] and Nicolas Palix [42]. They both defended at the beginning of this evaluation period.

Avionics. Safety-critical applications have to fulfill stringent requirements, both functional and non-functional. These requirements have to be coherent with each other and must be preserved throughout the software development process. In this context, a design-driven development approach can play a critical role. However existing design-driven development approaches are often general purpose, providing little, if any, conceptual framework to guide the development. Previously, we explained how the DiaSuite approach was enriched with non-functional declarations such as QoS and error handing.

To validate the interest of DiaSuite for safety-critical applications, several avionics case studies have been realized in the context of a collaboration with Thales, a French airborne systems company. One case study was a flight guidance application; it is in charge of the plane navigation and is under the supervision of the pilot. For example, if the pilot specifies a heading to follow, the application compares it to the current heading, sensed by devices such as the Inertial Reference Unit, and maneuvers ailerons accordingly. To test this application, we have used the DiaSim tool coupled with the FlightGear simulator. A flight guidance application has also been developed for a commercial drone platform. The goal of this application was to make the drone autonomous by following a flight plan similar to the one in avionics.

This simulation work has been presented in the thesis of Julien Bruneau [11]. Non-functional concerns addressing error handling and QoS will be presented in two forthcoming dissertations.

6.3.2. DiaSuiteBox: an Ongoing Technology-Transfer Project.

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

DiaSuiteBox consists of three main components:

- A tool-based environment is dedicated to the development of applications, orchestrating networked entities. This environment leverages DiaSpec, its compiler and an Eclipse plugin.
- An application store is composed of two servers: (1) a server verifies and packages submitted applications of developers prior to making them available to users and (2) another server enables users to browse, select and install applications.
- An execution environment runs end-user applications and allows to manage and configure all aspects of a smart space. This environment can either be deployed on low-resource computing platform (*e.g.,* Plug-PC, set-top-box) at the end-user's home or in the Cloud, coupled with a gateway for controlling equipments on the end-user's side.

Thanks to the application store and a developer community, the platform should provide users with a stream of innovative applications. During the submission process, an application is automatically analyzed and checked in order to be certified. [3] The user is ensured that the behavior of its applications is innocuous and conform to their description. DiaSuiteBox supports several technology standards like UPnP, Bluetooth, USB...This platform can be easily extended by plugging appliances directly on the hardware platform or by connecting devices on the local network.

7. Bilateral Contracts and Grants with Industry

7.1. Bilateral Grants with Industry

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

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

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

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

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

8. Partnerships and Cooperations

8.1. Regional Initiatives

8.1.1. Assistive Technologies for Elderly

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

[3] This certification process is preliminary in the current version of DiaSuiteBox.

This work is funded by CARSAT Aquitaine ("Caisse d'Assurance Retraite et de la Santé au Travail").

8.1.2. Cognitive Assistance for Supporting the Autonomy of Persons with Intellectual Disabilities

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

8.1.3. Certification of an open platform

The purpose of this project is to define concepts and tools for developing certifying open platforms. This certification process must ensure a set of critical properties (*e.g.,* safety, confidentiality, security) by certifying each tier application. These guarantees are essential to ensure that openness does not come at the expense of the user's well-being. To preserve the innovation model of open platforms, this certification process should also be as automatic as possible. Indeed, the success of open platforms is mainly due to the low development cost of a new application. The case study of this thesis will be the domain of home automation. The results of this thesis will be put into practice in the DiaSuiteBox open platform.

This project is funded by the Aquitaine region.

8.2. National Initiatives

8.2.1. Objects' World: design-driven development of large-scale smart spaces

The goal of this project is to develop an innovative communication technology, allowing the emergence of a new economic sector for large-scale smart spaces. Our objective is to propose concepts and tools for developing reliable applications orchestrating large-scale smart spaces of networked entities. The industrial partners of the Objects' World project will provide us with real-size case studies in various application domains (e.g., smart cities, tracking of vehicles, healthcare, energy management).

This work is funded by the OSEO national agency.

8.2.2. SERUS: Software Engineering for Resilient Ubiquitous Systems

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

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

8.2.3. School Inclusion for Children with Autism

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

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

8.3. European Initiatives

8.3.1. Collaborations in European Programs, except FP7

Program: SUDOE territorial cooperation program (Interreg IV B)
Project acronym: Biomasud
Project title: Mechanisms for sustainability and enhancement of solid biomass market in the space of SUDOE
Duration: July 2011 - June 2013
Coordinator: AVEBIOM
Other partners: UCE (Consumers Union of Spain), CIEMAT (Public Research Agency for excellence in energy and environment, Spain), CBE (Centro da Biomassa para a Energia, Portugal), CVR (Centro para la Valorización de Residuos, Portugal) and UCFF (Union Française de la Coopération Forestière, France)
Abstract: The goal of the Biomasud european project is to show the viability of the biomass-based energy model. The project aims to propose a certification and traceability process throughout the value chain of biofuel. Our objective is to design and implement a prototype of traceability system that will extract automatically traceability information based on sensors such as RFID tags, simplifying the certification process. This work will leverage our DIASUITE development methodology and will be evaluated by the Biomasud partners.

8.4. International Initiatives

8.4.1. Inria International Partners

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

8.5. International Research Visitors

8.5.1. Visits of International Scientists

The Phoenix group has been visited by Tim Sheard for 3 months (January-March).

8.5.2. Visits to International Teams

Charles Consel is on sabbatical for the academic year of 2012-2013 at the University of Mc Gill in Montreal.

9. Dissemination

9.1. Scientific Animation

Charles Consel has been involved in the following events as

- Program Committee member of
  - NIER track of ICSE 2012 (co-chair): New Ideas and Emerging Results track of the 34th International Conference on Software Engineering.
  - IEEE ICWS 2012: the 19th International Conference on Web Services
  - WEH 2012: ICSE International Workshop on Exception Handling
- Invited speaker at
  - Programming Language Mentoring Workshop, POPL 2012
  - Colloquium series of the department of computer science of Halmstad University, Sweden, June 2012.
  - Colloquium series of the department of Computer Science, University of Montreal, Canada, October 2012.
  - Colloquium series of the department of Computer Science, Mc Gill University, Canada, December 2012.
  - Colloquium series of the department of Computer Science, University of Sherbrooke, Canada, December 2012.

Charles Consel has participated in the following thesis defense committees:

- Julien Bruneau, May 16, University of Bordeaux, France
- Hongyu Guan, June 1, University of Bordeaux, France

Emilie Balland has been involved in the following events as

- Program Committee member of
  - SLE 2012: 5th International Conference on Software Language Engineering
  - WRLA 2012: 9th International Workshop on Rewriting Logic and its Applications (Co-located with ETAPS 2012)
  - PEPM 2012: ACM SIGPLAN 2012 Workshop on Partial Evaluation and Program Manipulation (Co-located with POPL 2012)
  - LDTA 2012 (local chair): 12th Workshop on Language Descriptions, Tools and Applications (Co-located with ETAPS 2012)
- Member of the thesis defense committee of Julien Bruneau, May 16, University of Bordeaux

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

Master level courses:

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

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

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

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

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

Master : Emilie Balland, Software Development Methodologies, 40 hours (M2 level), Ho Chi Minh University, Vietnam.

9.2.2. Supervision

PhD : Julien Bruneau, “Plateforme d’exécution paramétrable de systèmes communicants”, University of Bordeaux, May 16, 2012, supervised by Charles Consel

PhD : Hongyu Guan, “Gestion de l’hétérogénéité des environnements ubiquitaires et de la consommation d’énergie des environnements mobiles”, University of Bordeaux,June 1, 2012, supervised by Charles Consel

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

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

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

9.2.3. Juries

Charles Consel was a member of the selection committee for a professor position at Rennes University (section 27).

9.3. Popularization

- Demonstration of DiaSuiteBox during Rencontre Inria Industries “Sciences Numériques et Efficacité Énergétique” (digital technologies for efficient energy use), March 2012
- General audience article in the Sud-Ouest newspaper, 15/03/12

10. Bibliography

Major publications by the team in recent years

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

[2] D. CASSOU, J. BRUNEAU, C. CONSEL, E. BALLAND. *Towards a Tool-based Development Methodology for Pervasive Computing Applications*, in "IEEE TSE: Transactions on Software Engineering", 2012, vol. 38, n^{o} 6, p. 1445-1463, http://hal.inria.fr/hal-00683210.

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

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

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

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

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

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

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

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

Publications of the year

Doctoral Dissertations and Habilitation Theses

[11] J. BRUNEAU. *Developing and Testing Pervasive Computing Applications: A Tool-Based Methodology*, Université Sciences et Technologies - Bordeaux I, May 2012, http://tel.archives-ouvertes.fr/tel-00767395.

[12] H. GUAN. *Canevas de programmation pour gérer l'héterogénéité et la consommation d'énergie des mobiles dans un environnement ubiquitaire*, Université Sciences et Technologies - Bordeaux I, June 2012, http://hal.inria.fr/tel-00719175.

Articles in International Peer-Reviewed Journals

[13] E. BALLAND, P.-E. MOREAU, A. REILLES. *Effective Strategic Programming for Java Developers*, in "Software: Practice and Experience", October 2012, (To appear).

[14] B. BERTRAN, J. BRUNEAU, D. CASSOU, N. LORIANT, E. BALLAND, C. CONSEL. *DiaSuite: a Tool Suite To Develop Sense/Compute/Control Applications*, in "Science of Computer Programming, Fourth special issue on Experimental Software and Toolkits", 2012 [*DOI :* 10.1016/J.SCICO.2012.04.001], http://hal.inria.fr/hal-00702909.

[15] J. BRUNEAU, C. CONSEL. *DiaSim: A Simulator for Pervasive Computing Applications*, in "Software: Practice and Experience", 2012, http://hal.inria.fr/hal-00715745.

[16] D. CASSOU, J. BRUNEAU, C. CONSEL, E. BALLAND. *Towards a Tool-based Development Methodology for Pervasive Computing Applications*, in "IEEE TSE: Transactions on Software Engineering", 2012, vol. 38, n^{o} 6, p. 1445-1463, http://hal.inria.fr/hal-00683210.

[17] Z. DREY, C. CONSEL. *Taxonomy-Driven Prototyping of Home Automation Applications : a Novice-Programmer Visual Language and its Evaluation*, in "Journal of Visual Languages and Computing", December 2012, (To appear), http://hal.inria.fr/hal-00718943.

International Conferences with Proceedings

[18] J. BRUNEAU, C. CONSEL, M. O'MALLEY, W. TAHA, W. M. HANNOURAH. *Virtual Testing for Smart Buildings*, in "IE 2012 - 8th International Conference on Intelligent Environments", Guanajuato, Mexico, June 2012, http://hal.inria.fr/hal-00715753.

[19] Q. ENARD, C. LOUBERRY, C. CONSEL, X. BLANC. *An Experimental Study of A Design-driven, Tool-based Development Approach*, in "User Evaluation for Software Engineering Researchers (USER)", Zurich, Switzerland, 2012 [*DOI :* 10.1109/USER.2012.6226581], http://hal.inria.fr/hal-00715759.

References in notes

[20] J. ALDRICH, V. KOSTADINOV, C. CHAMBERS. *Alias Annotations for Program Understanding*, in "OOPSLA'02: Proceedings of the 17th International Conference on Object-Oriented Programming, Systems, Languages, and Applications", New York, NY, USA, ACM, 2002, vol. 37(11), p. 311–330.

[21] P. AMEY. *Correctness by Construction: Better Can Also Be Cheaper*, in "CrossTalk: the Journal of Defense Software Engineering", 2002, vol. 2, p. 24–28.

[22] A. AVIZIENIS, J. LAPRIE, B. RANDELL, C. LANDWEHR. *Basic Concepts and Taxonomy of Dependable and Secure Computing*, in "Dependable and Secure Computing, IEEE Transactions on", 2004, vol. 1, n^{o} 1, p. 11–33.

[23] J. BRUNEAU, C. CONSEL, M. O'MALLEY, W. TAHA, W. M. HANNOURAH. *Preliminary Results in Virtual Testing for Smart Buildings (Poster)*, in "MOBIQUITOUS 2010, 7th International ICST Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services", Sydney, Australia, December 2010, http://hal.inria.fr/inria-00551264.

[24] D. CASSOU, J. BRUNEAU, C. CONSEL. *A Tool Suite to Prototype Pervasive Computing Applications (Demo)*, in "Proceedings of the 8th IEEE Conference on Pervasive Computing and Communications (PERCOM'10)", Mannheim, Germany, IEEE Computer Society Press, 2010, p. 1–3, http://hal.inria.fr/inria-00484067.

[25] D. CASSOU, J. BRUNEAU, J. MERCADAL, Q. ENARD, E. BALLAND, N. LORIANT, C. CONSEL. *Towards a Tool-based Development Methodology for Sense/Compute/Control Applications (Poster)*, in "SPLASH'10: Proceedings of the 1st International Conference on Systems, Programming, Languages, and Applications: Software for Humanity", Reno/Tahoe, United States, ACM, 2010, p. 1–2, poster, http://hal.inria.fr/inria-00510378.

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

[27] C. CONSEL, H. HAMDI, L. RÉVEILLÈRE, L. SINGARAVELU, H. YU, C. PU. *Spidle: A DSL approach to specifying streaming application*, in "International Conference on Generative Programming and Component Engineering", Germany, 2003, p. 1-17, http://hal.inria.fr/hal-00350193.

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

[29] C. CONSEL, L. RÉVEILLÈRE. *A programmable client-server model: Robust extensibility via DSLs*, in "International Conference on Automated Software Engineering", Montréal, Canada, 2003, p. 70-79, http://hal.inria.fr/hal-00350051.

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

[31] P. FEILER. *The Architecture Analysis & Design Language (AADL): An Introduction*, DTIC Document, 2006.

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

[33] J. GRAY, K. FISHER, C. CONSEL, G. KARSAI, M. MERNIK, J.-P. TOLVANEN. *DSLs: the good, the bad, and the ugly*, in "Conference on Object Oriented Programming Systems Languages and Applications archive", Nashville, United States, ACM, 2008, http://hal.inria.fr/inria-00402566.

[34] H. JAKOB, C. CONSEL, N. LORIANT. *Architecturing Conflict Handling of Pervasive Computing Resources*, in "11th IFIP International Conference on Distributed Applications and Interoperable Systems (DAIS'11)", Reykjavik, Iceland, June 2011, http://hal.inria.fr/inria-00581604.

[35] W. JOUVE, J. BRUNEAU, C. CONSEL. *DiaSim: A Parameterized Simulator for Pervasive Computing Applications (Demo)*, in "IEEE International Conference on Pervasive Computing and Communications, 2009", Galveston, United States, March 2009, http://hal.inria.fr/inria-00441351.

[36] W. JOUVE. *Approche déclarative pour la génération de canevas logiciels dédiés à l'informatique ubiquitaire*, Université Sciences et Technologies - Bordeaux I, April 2009, http://hal.inria.fr/tel-00402605.

[37] G. LASNIER, B. ZALILA, L. PAUTET, J. HUGUES. *OCARINA: An Environment for AADL Models Analysis and Automatic Code Generation for High Integrity Applications*, in "Reliable Software Technologies–Ada-Europe", 2009, p. 237–250.

[38] F. LATRY, J. MERCADAL, C. CONSEL. *Staging Telephony Service Creation: A Language Approach*, in "Principles, Systems and Applications of IP Telecommunications", New-York, United States, 2007, http://hal.inria.fr/inria-00353534.

[39] B. LITTLEWOOD, L. STRIGINI. *Software Reliability and Dependability: a Roadmap*, in "ICSE'00: Proceedings of the Conference on The Future of Software Engineering", ACM, 2000, p. 175–188.

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

[41] 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.

[42] N. PALIX. *Langages dédiés au développement de services de communications*, Université Sciences et Technologies - Bordeaux I, September 2008, http://hal.inria.fr/tel-00340864.

[43] N. PALIX, L. RÉVEILLÈRE, C. CONSEL, J. LAWALL. *A Stepwise Approach to Developing Languages for SIP Telephony Service Creation*, in "Proceedings of Principles, Systems and Applications of IP Telecommunications, IPTComm", New York City, United States, ACM Press, 2007, p. 79-88, http://hal.inria.fr/inria-00196520.

[44] A. R. PERRY. *The FlightGear Flight Simulator*, in "Proceedings of the USENIX Annual Technical Conference", 2004.

[45] 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.

[46] 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.

[47] M. SHAW. *Beyond Objects: A Software Design Paradigm Based on Process Control*, in "SIGSOFT Software Engineering Notes", January 1995, vol. 20, p. 27–38.

[48] R. N. TAYLOR, N. MEDVIDOVIC, E. M. DASHOFY. *Software Architecture: Foundations, Theory, and Practice*, Wiley, 2009.

[49] S. THIBAULT, C. CONSEL, G. MULLER. *Safe and efficient active network programming*, in "Reliable Distributed Systems, 1998. Proceedings. Seventeenth IEEE Symposium on", IEEE, 1998, p. 135–143.

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

[51] N. UBAYASHI, J. NOMURA, T. TAMAI. *Archface: A Contract Place Where Architectural Design and Code Meet Together*, in "ICSE'10: Proceedings of the 32nd International Conference on Software Engineering", New York, NY, USA, ACM, 2010, p. 75–84.

[52] S. VESTAL. *An Overview of the Architecture Analysis & Design Language (AADL) Error Model Annex*, in "AADL Workshop", 2005.

[53] M. VOLTER, T. STAHL, J. BETTIN, A. HAASE, S. HELSEN. *Model-driven Software Development: Technology, Engineering, Management*, John Wiley and Sons Ltd, 2006.