INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Sardes

System Architecture for Reflective Distributed Computing Environments

Rhône-Alpes

THEME COM

Activity Report

2006

Table of contents

1. Team

SARDES *is a project of Inria Rhône-Alpes and a research team of Imag-LSR (Software, Systems and Networks Laboratory), a joint research unit (UMR 5526) of Centre National de la Recherche Scientifique, Institut National Polytechnique de Grenoble and université Joseph Fourier.*

Head of Project-team
Jean-Bernard Stefani [Inria Research Director, on leave from *corps des Télécommunications*, HdR]

Administrative Assistant
Élodie Toihein

Inria Personnel
Alan Schmitt [research scientist]

Other Permanent Personnel
Sara Bouchenak [associate professor, université Joseph Fourier]
Fabienne Boyer [associate professor, université Joseph Fourier]
Noël De Palma [associate professor, Institut National Polytechnique de Grenoble]
Sébastien Jean [associate professor, université Pierre Mendès France]
Sacha Krakowiak [professor, université Joseph Fourier, HdR]
Renaud Lachaize [associate professor, université Joseph Fourier, from Oct. 1, 2006]
Jacques Mossière [professor, Institut National Polytechnique de Grenoble, HdR]
Vivien Quéma [research scientist, Centre National de la Recherche Scientifique, from Oct. 1, 2006]

Post-doctoral Fellows
Adrian Mos [University College, Dublin, Ireland, till Aug. 31, 2006]
Vivien Quéma [from July 15 to September 30, 2006]

Visitor
Didier Donsez [université Joseph Fourier, from Sep. 1, 2006]

Technical Staff
Julien Legrand
Pierre Garcia [from July 1, 2006]
Florent Métral [from July 1, 2006]
Nikos Parlavantzas [from Nov 9, 2006]

Ph.D. Students
Takoua Abdellatif [industrial grant (Bull), till Sep. 15, 2006]
Benoît Claudel [government grant, from Oct. 1, 2006]
Stéphane Fontaine [government grant, from Oct. 1, 2006]
Jakub Kornaś [Inria grant (contract)]
Michaël Lienhardt [from Oct. 1, 2006]
Sergueï Lenglet [from Oct. 1, 2006]
Ali Erdem Özcan [industrial grant (STMicroelectronics)]
Juraj Polakovic [France Telecom R&D]
Jérémy Philippe [Inria grant (contract)]
Sylvain Sicard [Inria grant (contract)]
Christophe Taton [government grant]

Interns
Benoît Claudel [Master's student, École Doctorale de Grenoble, till Sep. 30, 2006]
Stéphane Fontaine [Master's student, École Doctorale de Grenoble, till Sep. 30, 2006]
Michaël Lienhardt [4th year student at École Normale Supérieure de Cachan, till Sep. 30, 2006]
Sergueï Lenglet [Master's student, 3rd year student at École Normale Supérieure de Lyon, till Sep. 30, 2006]
Jean Arnaud [Master's student, École Doctorale de Grenoble, from Oct. 1, 2006]
Fabien Gaud [Master's student, École Doctorale de Grenoble, from Oct. 1, 2006]

Sylvain Gonzalez [M. Eng. intern, Conservatoire National des Arts et Métiers, from Oct. 1, 2006]
Willy Malvault [Master's student, École Doctorale de Grenoble, from Oct. 1, 2006]
Brice Saccucci [Master's student, École Doctorale de Grenoble, from Oct. 1, 2006]

2. Overall Objectives

2.1. Goals and Organization

The overall goal of the SARDES project is to develop concepts, methods and tools to build distributed systems (primarily, software infrastructure - i.e. operating systems and middleware) that are open, evolvable and reliable. The designers and developers of such systems are facing considerable challenges, among which dynamicity, scalability, system management complexity, and quality of service. To respond to these challenges, distributed systems need to be highly adaptable and self-manageable.

The project has two main ambitions:

1. To develop rigorous programming models and supporting software engineering techniques to build efficient, adaptable distributed systems.
2. To apply the concepts, models and techniques developed in the project to the construction of self-manageable systems.

In line with these ambitions, the project conducts research in several areas:

- Component models and frameworks. We believe that the key to building flexible and adaptable systems is to develop a provably sound and efficient reflective software component technology based on a formal (mostly operational) semantical basis. The FRACTAL component model is the base of our work on reflective component technology. It is a general component model supporting hierarchical and dynamic composition with sharing, founded on a mathematical base, the Kell calculus.
- Programming languages. In order to simplify the work of programmers, we design programming languages and tools, such as type systems, adapted to the specific problems we want to address. More precisely, we develop programming languages for component assemblages and bidirectional data manipulation, and type systems for correct manipulation of messages.
- Distributed algorithms. Many algorithms that are considered optimal in theory fail to live up to their potential in practice. Our goal is to help closing this gap between theory and practice by proposing complexity models closer to reality and by revisiting classical abstractions considering these new models. We are currently applying this approach to the design of algorithms for total order broadcast, eventual consistency, distributed shared memories, and publish/subscribe systems over P2P networks.
- Flexible operating systems kernels. Despite years of research and practice, developing (or tailoring) an operating system for the needs of a given application domain remains a very complex process. We are therefore investigating new techniques to ease the development of robust and adaptable kernels. Our work mostly focuses on architectural patterns, programming models and optimized algorithms for embedded systems such as multimedia devices and sensor networks.
- Autonomic systems. Autonomic computing aims at providing systems and applications with self-management capabilities, including self-configuration (automatic configuration according to a specified policy), self-optimization (continuous performance monitoring), self-healing (detecting defects and failures, and taking corrective actions), and self-protection (taking preventive measures and defending against malicious attacks). For the design of the feedback control loops that achieve these functions, we propose an architecture-based approach, i.e. one based on a semi-formal description of the structure of the controlled system using a reflective, dynamically configurable, component model.

This research is validated through the design, construction and evaluation of several prototype systems, some of which form a base for collaboration with industrial partners.

- DREAM: a framework for flexible communication systems. Dream is a component-based framework dedicated to the construction of communication middleware. It provides a component library and a set of tools to build, configure and deploy middleware implementing various communication paradigms: group communications, message passing, event-reaction, publish-subscribe, etc. DREAM builds upon the FRACTAL component framework, which provides support for hierarchical and dynamic composition.
- JADE: a framework for autonomic management. We are developing a framework, called JADE, for the construction of autonomic systems using our reflective component model. The controlled system is described in terms of an assembly of components equipped with elementary management capabilities. This description, in turn, is the base of the feedback control loops that implement the self-management functions described above. Legacy applications are managed by wrapping them into components. Since JADE is itself developed using the component model, the autonomic functions also apply to JADE.
- THINK: a framework for flexible kernels. THINK is a component-based programming framework for building modular and customizable operating systems and applications. THINK can be considered as an implementation of the FRACTAL component model that enables component-based programming in the C language. It is being used for the development of flexible kernels for Systems on Chips (SoC).

In return, the development of these systems poses new fundamental questions in the research areas described above.

2.2. Collaborations and Personnel Flow

SARDES is involved in several industrial and international collaborations. It is active in the OBJECTWEB consortium (8.2) dedicated to open source middleware. It is a partner of several European projects and networks: CoreGrid, Gorda, SelfMan and Grid4All (IST program); S4All (ITEA Eureka program). It actively participates in the national research network on software technologies (RNTL), with two new projets (SelfWare and JOnES) started in 2006 and two more (Flex-e-ware and ScorWare) already selected for 2007. It participates in other national research initiatives : Modyfiable (ARA program) and AutoMan (INRIA internal collaborative action). It collaborates with several industrial partners: Bull, France Telecom, STMicroElectronics, and has close links with Scalagent, a technology startup created by former members of the project.

Note that two new permanent members, Renaud Lachaize and Vivien Quéma, have joined the project team in 2006.

3. Scientific Foundations

3.1. Introduction

In this section, we first present the main challenges that face the designers of large scale distributed systems (3.2). We then discuss recent advances and open problems in two areas covered by SARDES: component-based architectures for adaptable distributed systems (3.3), and distributed system management (3.4).

3.2. Challenges of Distributed Systems

The future[1] of information processing applications is envisioned as a range of environments in which processors will be everywhere and will be interconnected by an array of networks, from ad-hoc to the global Internet. Constructing the software base - the *middleware* - for such ubiquitous computing infrastructures poses a number of scientific and technical challenges, which arise from the wide variety of applications and services that need to be supported.

Software will run in a multitude of computing environments ranging from traditional desktops to multiple host, networked systems, to mobile systems, to embedded systems, to wearable computers. Software systems will need to be "aware" of their physical surroundings, taking input from real-world sensors and sending output intended to control the real-world environment. They will need to "understand" their proximity to other computing resources and use this information to guide their behavior. A number of application scenarios illustrating these new environments have been discussed in a recent report for the IST European program [47].

A fundamental requirement for such environments is for middleware architectures capable of supporting services that are *composable* and *adaptable*. We have singled out four major challenges: scalability, quality of service, manageability and programmability.

1. *Scalability* concerns the ability to scale in several dimensions; scaling in machine forms: from smart labels to server farms to meta-computing network overlays; scaling in numbers: objects, machines, users, locations; scaling in logical and organizational structures: from ad-hoc collaborations networks to federations of multi-domain enterprises.
2. *Quality of service (QoS)* concerns the ability to obtain correctness and service-level guarantees such as timeliness, availability, fault-tolerance, survivability for applications executing in large scale environments. The problem of meeting QoS requirements of applications is made harder in a ubiquitous computing environment where new services and customized services are expected to be added into (existing) applications at an alarming rate.
3. *Manageability* concerns the ability to monitor and to control the operation and evolution of large scale, long-lived distributed applications and services, avoiding manual intervention and centralization (i.e., assumption of one management domain). Distributed applications and services will need to be reconfigured dynamically, for example, to maintain user specified QoS guarantees despite changes in operating conditions (e.g., component failures). Mechanisms are needed to dynamically add, extend, remove or move component services in a dependable and predictable manner.
4. *Programmability* concerns the ability to compose new applications from existing applications and services, to deploy and to maintain them in highly dynamic and heterogeneous computing environments; further, applications are expected to be highly parallel, requiring high-level abstractions necessary for dealing with complexity, with fine-grained resource awareness built according to the end-to-end principle.

Middleware for a ubiquitous computing environment will by necessity be open and standardized, at least at the level of communication protocols, key application programming interfaces and language tools. Furthermore, such middleware will become an economic commodity, i.e. be available at very low or no cost in all manners of equipment and networks to serve as a viable basis for value-added services and applications. In this context, it makes both technical and economical sense to pursue the development of such middleware facilities using an open source model.

In the next two sections, we discuss specific aspects relevant to the current research interests of SARDES.

3.3. Architectures for Adaptable Distributed Systems

[1]The text of 3.2 has been adapted from a Research Roadmap Report produced by the Midas IST project in which SARDES has participated.

A first requirement for adaptable middleware is a modular architecture, in which the interfaces and dependencies between the parts should be clearly identified. The main motivations are to minimize the adaptation induced changes by defining components as adaptation localities, and to use structural modifications (e.g. reconfiguration) as a tool for adaptation. Thus research on adaptable middleware is essentially done in the context of component-based systems. This is the theme of Section 3.3.1.

Our approach to the design and implementation of adaptable, component-based systems is presented in Section 3.3.2

3.3.1. Adaptable Component-based Middleware

Building an application out of composable parts is an old challenge of software engineering. Despite the apparent simplicity of its requirements, progress towards this goal has been slow. The notions related to components have only been elicited recently [58]. Although commercial infrastructures based on components are now available (e.g. COM+, CCM, EJB, .Net), the rigorous definition of a component model and of its architectural invariants is still an open issue.

Three main approaches are being used to achieve adaptability and separation of concerns in middleware systems: meta-object protocols, aspect-oriented programming, and pragmatic techniques.

Meta-object protocols [53] are the basic device of reflection. A reflective system is one that is able to answer questions about itself and to modify its own behavior, by providing a causally connected representation of itself to a meta-level. In a component infrastructure, the interface available at the meta-level allows the operations related to component lifecycle and composition to be dynamically adapted. A number of research prototypes (OpenORB [48], Flexinet, 2K/DynamicTAO) have been developed to investigate this approach.

The goal of aspect-oriented programming [54] (AOP) is to define methods and tools to better identify and isolate the code related to the various “aspects” present in an application, such as those related to extra-functional properties (security, fault tolerance, performance). This code is usually intertwined with that of the application proper, which is detrimental to easy change.

Pragmatic techniques usually rely on interception (interposing code at the interface between components, while preserving the interface). They are mainly used in commercial middleware.

The price of the increased flexibility brought by these adaptation techniques is paid in performance, due to the additional indirection or interpretation levels that these techniques introduce. Thus optimization techniques for reducing this performance penalty are an active subject of research. They include partial evaluation and program specialization [55], code injection (inlining), dynamic code generation and optimization [57].

3.3.2. Our Approach to Component-based Systems

The main goal in the Reflective Component Technology theme in SARDES is to develop a provably sound and efficient reflective software component technology for the construction of highly reconfigurable distributed systems, and in particular distributed software infrastructures (operating systems and middleware).

The approach taken in this theme combines a component-based approach to software construction with reflective techniques to serve as a basis of a component-based distributed programming model. In the approach, components can be used as units of encapsulation (e.g. for the purposes of modular system construction and fault isolation), and units of configuration (e.g. for the purposes of deployment, dynamic update and adaptation). Reflective components, i.e. components that expose meta-object protocol interfaces to allow for instrospection and intercession, provide the basis for constructing dynamically configurable systems.

Work on this theme places a strong emphasis on developing programming models with a formal (mostly operational) semantical basis. We believe this is essential both to ensure a sound design, but also to serve as a basis for the construction of provably safe and secure systems.

Our reflective component-based approach to software systems construction combines with an overall “exo-kernel” philosophy for the design of software infrastructures. In essence, this design philosophy emphasizes the need to refrain from building software infrastructures with too many pre-determined abstractions and pre-defined services, in order to maximize flexibility and configurability in said infrastructures. The component model that embodies these principles is FRACTAL [13], jointly developed with France Telecom R&D. It allows unlimited hierarchical composition with sharing (i.e. a component may be shared between several enclosing components). A component has a control interface that allows managing its properties (lifecycle, attributes, contents, bindings) through appropriate controllers. These controllers may themselves be adapted and extended.

This theme includes the following research activities.

1. Models and foundations.

   In this work, the aim is to develop novel models for component-based distributed programming, grounded on a rigourous semantical basis. The approach is to exploit and develop techniques from process calculi (e.g. behavioral equivalences, type systems) to formally characterize the main primitives in our programming models, and to serve as a basis for establishing the correctness of supporting tools (such as abstract machines, type checkers or code generation tools).

   The main scientific challenges that we address, in this area, include: support for behavioral reflection, bisimulation-based theory for higher-order calculi, dependent type systems for higher-order calculi with localities, support for handling failures and recoverable activities, provably correct abstract machines for high-order component calculi, resource-aware components, and multi-stage programming.

2. Support for component-based distributed programming.

   In this work, the aim is to prototype basic support for component-based distributed programming. Ongoing investigations include the continued development of, and extensions to, the FRACTAL reflective component model, the definition of dynamic Architecture Description Languages (dynamic ADLs, i.e. ADLs that support the specification of component behavior and the description of reactive reconfigurations), the development of retargettable code generation tools for dynamic ADLs, the use of type systems to enforce component encapsulation constraints, and the development of optimization techniques such as proxy in-lining for run-time component structures.

   The main scientific challenges that we address, in this area, include: type systems and effective type checking for dynamic component structures, retargettable code generation for dynamic ADLs, reversible techniques for on-line component optimization, support for new target hardware architectures.

3.4. Autonomous Distributed System Management

Section 3.4.1 introduces and motivates the new area known as *autonomic computing*. Section 3.4.2 presents our approach to research in this area.

3.4.1. Motivations for Autonomic Computing

Management (or *Administration*) is the function that aims at maintaining a system's ability to provide its specified services, with a prescribed quality of service. In general terms, administration may be viewed as a *control* activity, involving an event-reaction loop: the administration system detects events that may alter the ability of the administered system to perform its function, and reacts to these events by trying to restore this ability. The operations performed under system and application administration include configuration and deployment, reconfiguration, resource management, observation and monitoring.

Up to now, administration tasks have mainly been performed by persons. A great deal of the knowledge needed for administration tasks is not formalized and is part of the administrators' know-how and experience. As the size and complexity of the systems and applications are increasing, the costs related to administration are taking up a major part of the total information processing budgets, and the difficulty of the administration tasks tends to approach the limits of the administrators' skills. For example, an analysis of the causes of failures of Internet services [56] shows that most of the service's downtime may be attributed to management errors (e.g. wrong configuration), and that software failures come second. In the same vein, unexpected variations of the load are difficult to manage, since they require short reaction times, which human administrators are not able to achieve. The traditional approach based on the manager-agent model, the base of widely used management protocols and frameworks such as SNMP and CMIS/CMIP, is also showing its inability to cope with the current highly dynamic managed systems.

The above remarks have motivated a new approach, in which a significant part of management-related functions is performed automatically, with minimal human intervention. This is the goal of the so-called *autonomic computing* movement [51].

In the area of system administration, the role of configuration and deployment has long been neglected. Current research [49], [50] aims at giving a rigorous foundation to these phases of the lifecycle, thus reducing the likelihood of misconfiguration, and opening the way to autonomous configuration and deployment.

Several research projects [45] are active in this area. [52] is a recent survey of the main research problems related to autonomic computing.

3.4.2. Our Approach to Autonomous Systems Management

The main goal in the Autonomous Distributed System Management theme in SARDES is to develop component-based software infrastructure frameworks and tools for the control and administration of large scale, long-lived distributed systems, such as clustered application servers or Grid systems. Target application domains for our prototype autonomous management systems include large Web application servers such as J2EE servers, large information mediation servers such as enterprise service buses (ESBs), and Grids.

This theme includes the following research activities.

1. Infrastructure for system management.

   In this work, the aim is to develop tools and middleware technology to support the principled implementation of system management loops. We consider the following subjects.

   – *Instrumentation for component structures:* the aim is to develop a set of tools complementing our component-based software engineering tools with specific low-level logging, monitoring and resource accounting functions. This should provide us with the technology required to efficiently and dynamically implement sensors for our systems control and management loops.

   – *Asynchronous middleware:* the aim is to develop a dynamically configurable and scalable technology for the construction of large scale (number of events, number of components, geographical dispersion) asynchronous information dissemination channels. Such channels are crucial in system management for the efficient transport of event notifications.

     Challenges in this area include: devising scalable routing schemes for event dissemination, defining composable abstractions for the construction of multiple asynchronous middleware personalities, fault-tolerance and overload management.

   – *Dynamic system cartography:* the aim is to develop a multi-level, multi-grain monitoring services, able to construct and maintain during a system or application life-time, a causally connected view of the system or application being managed.

2. Distributed configuration management.

   In this work, the aim is to develop a comprehensive set of system/software deployment and configuration services, from low-level bootstrapping services for component loading and system initialization, to higher-level routing, scheduling and orchestration for complex distributed system on-line deployment and configuration, with multiple component bindings and dependencies.

   Challenges in this area include: automatic resources and services discovery, dealing with partial failures, dealing with multiple dependencies and multiple coexisting versions, controlling and optimizing configuration and deployment workflows, automated support for high-level configuration policies.

3. Autonomic capabilities.

In this work, the aim is to study the automation of various systems management functions, and to allow in particular automated performability (i.e. the combination of performance and availability) management under quality of service and service level agreement constraints. The approach we follow as a first step is both *architecture-based*, i.e. leveraging an explicit component-based structure at run-time, and *empirical*, i.e. relying mostly on the empirical derivation of performance and availability models through the instrumentation and run-time monitoring of specific experimental systems. In a second step, we plan to investigate the use of more sophisticated modelling tools and techniques, including e.g. the use of control theory techniques for deriving and synthesizing control laws for distributed resource management and overload management. Capabilities we consider initially include: automated repair management and automated sizing, starting with cluster-size systems such as clustered Web application servers.

Repair management is a direct complement to standard fault tolerance and fault recovery techniques in distributed systems, that allows a managed system to be brought back into a target regime of behavior, through reconfiguration and resource allocation, after the occurrence of a (non malicious) fault. Challenges in automating repair management include: dealing with a mixture of hardware and software faults, as well as different fault models; automatic fault detection and diagnosis; support for self-healing; support for repair management in large scale distributed systems.

Automated sizing or self-sizing aims to adapt automatically the set of resources used by the provider of a service to the level of demand for this service, e.g. automatically acquiring or releasing resources for a Web application server, according to the load of client requests. Challenges in self-sizing include: load and overload characterization in a distributed system, identifying relevant control parameters for performance tuning, devising effective algorithms for distributed performance control and load balancing.

4. Application Domains

4.1. Links to Applications

SARDES develops generic tools for distributed applications, in the form of middleware, system kernels, and information servers. These tools are useful in application domains that have one or more of the following properties.

- Cooperation using shared distributed information;
- Mobility of users, information and services;
- Need for dynamic adaptation of infrastructures or applications;
- Use of high performance information servers.

Applications are important for a project like SARDES, in which experimental aspects play a significant part. They provide testbeds to evaluate prospective designs, and they help us establish links with industrial partners, allowing us to transfer results and to identify relevant research problems.

4.2. Current Application Areas

Keywords: *electronic commerce*, *embedded systems*, *multimedia*, *power supply*, *systems administration*, *telecommunications*.

In recent years, SARDES has been active in the following application areas:

1. Telecommunications : administration of large scale networks, servers and caches for the Web, management of configurable added value services;
2. Power supply : administration and monitoring of power supply networked equipment, e.g. uninterrupted power supply units.
3. Embedded computing: development of custom made kernels for specific applications (robotics, real time), dynamically reconfigurable kernels;
4. Multimedia applications: dynamic adaptation of a videoconferencing system for use by mobile clients;
5. Electronic commerce: flexible access to remote services by mobile users, efficient transaction management.

5. Software

5.1. Introduction

Keywords: *J2EE*, *autonomic computing*, *benchmark*, *cluster*, *clustered databases*, *communication middleware*, *publish-suscribe*, *transactions*.

Software development is an important aspect of the activity of SARDES. This software serves as a testbed to apply, validate and evaluate the methods and tools developed in the project.

Software developed in SARDES is (or will be) available in the OBJECTWEB open source code base (see 8.2), which is accessible at http://www.objectweb.org.

5.2. C-Jdbc: Clustered JDBC-accessed Database

Contact : see OBJECTWEB page.

JDBC™(short for Java DataBase Connectivity), is a widely used Java API for accessing virtually any kind of tabular data. C-JDBC (Clustered Java™DataBase Connectivity) is an open source database cluster middleware that allows any application to transparently access a cluster of databases through JDBC. The database is distributed and possibly replicated among several nodes and C-JDBC load balances the queries between these nodes.

C-JDBC is an OBJECTWEB project distributed under an LGPL license. C-JDBC has achieved a wide visibility, and has won the Derby Code Contest at ApacheCon US in November 2004.

See http://c-jdbc.objectweb.org

5.3. Dream, a Framework for Building Asynchronous Middleware

Contact : see OBJECTWEB page.

DREAM is a component-based framework dedicated to the construction of communication middleware. It provides a component library and a set of tools to build, configure and deploy middleware implementing various communication paradigms: group communications, message passing, event-reaction, publish-subscribe, etc. DREAM builds upon the FRACTAL component framework, which provides support for hierarchical and dynamic composition.

DREAM is an OBJECTWEB project distributed under an LGPL license.

See http://dream.objectweb.org

5.4. Jade, a Framework for Building Autonomic Management Systems

Contact : Noël De Palma.

JADE (see 6.6.1) is a framework for the construction of autonomic systems using the FRACTAL reflective component model. The controlled system is described in terms of an assembly of components equipped with elementary management capabilities. This description, in turn, is the base of the feedback control loops that implement various self-management functions. Legacy applications are managed by wrapping them into components. Since Jade is itself developed using the component model, the autonomic functions also apply to Jade.

JADE will be available under OBJECTWEB in 2007. In the meantime, contact noel.depalma@inrialpes.fr

5.5. Benchmarks for J2EE Systems

Contact : see OBJECTWEB page.

RUBIS (Rice University Bidding System)[2] is an auction site prototype modeled after `eBay.com`. RUBIS is currently used to evaluate design patterns, application servers and communication layers scalability. RUBIS has been implemented for PHP, Servlets and Enterprise Java Beans (EJB) in 7 different versions. Ongoing activities concern JDO (Java Data Objects) and Microsoft .Net versions of RUBIS. RUBBOS is a bulletin board benchmark modeled after an online news forum. Like RUBIS, RUBBOS was designed to evaluate design patterns, application servers and communication layers scalability.

Both RUBIS and RUBBOS are part of the OBJECTWEB JMOB (Java Middleware Open Benchmarking) project and are distributed under an LGPL license.

See http://jmob.objectweb.org

5.6. Jotm: Java Open Transactional Manager

Contact : see OBJECTWEB page.

JOTM is an open source transaction manager implemented in Java. It was originally part of the JONAS Enterprise Java Beans platform. Since 2002, it has been extracted from the JONAS environment and developed as an autonomous project to provide support for any distributed platform that needs transactional facilities.

JOTM implements the Java Transaction API (JTA) specification with full support on both RMI/JRMP and RMI/IIOP. It also implements the Business Transaction Protocol (BTP) to support transactions for Web Services.

JOTM is an OBJECTWEB project distributed under an LGPL license.

See http://jotm.objectweb.org

6. New Results

6.1. Introduction

In this section, we present new results in the main areas covered by SARDES: Component Models and Frameworks (6.2), Programming Languages (6.3), Distributed Algorithms (6.4), Flexible Operating Systems Kernels (6.5), and Autonomous Distributed System Management (6.6).

6.2. Component Models and Frameworks

Participants: Jean-Bernard Stefani, Alan Schmitt, Sergueï Lenglet, Michaël Lienhardt.

[2]Emmanuel Cecchet, a former member of Sardes, was a major contributor to RUBIS while he was at Rice University as a post-doctoral fellow.

The FRACTAL component model is the base of our work on reflective component technology. This section presents new results on the Kell calculus, the mathematical base for FRACTAL.

The M-calculus [10] has been introduced to overcome a number of deficiencies in the Distributed Join calculus, including the lack of control over migrating locations, the absence of dynamic binding, and the inability to support distributed locations (or components) with different semantics (e.g., with respect to failures or access control policies). The Kell calculus has in turn been introduced to alleviate some of the perceived complexity of the M-calculus (e.g., presence of multiple routing rules for communication) and to provide a more tractable setting for the development of a bisimulation theory. The type systems introduced for these calculi solve a non-trivial problem in higher-order process calculi with process passivation, namely that of ensuring the unicity of location names (a much harder problem than ensuring name linearity in the π-calculus).

Work on the component model has continued on two fronts during 2006. First, we have proposed a modeling of FRACTAL in the Kell calculus [35]. This modeling, partly based on previous work on sharing in the Kell calculus[3], aims at giving a precise semantics to component assemblages written in FRACTAL, and to allow the analysis of these programs using tools developed for the Kell calculus. Second, we have continued working on the challenging problem of bisimulations for the Kell calculus, in particular identifying the core difficulty as being the passivation operator [43]. We have for instance established that this difficulty is present in every calculus featuring passivation and is independent of the way scope extrusion is handled.

6.3. Programming Languages

Participants: Jean-Bernard Stefani, Alan Schmitt, Vivien Quéma, Michaël Lienhardt.

The following sections present new results in the area of problem-specific programming languages.

6.3.1. The OzK Language

Programming in a distributed and open environment remains challenging because it requires combining modularity, security, concurrency, distribution, and dynamicity. This has lead recently to interesting programming language developments such as Alice, Acute, Oz, JoCaml, ArchJava, etc. However, the combination of all the above features with dynamicity, i.e. the ability to build and modify systems during execution, still remains an open question. We propose an approach to open distributed programming that exploits the notion of locality, which has been studied intensively during the last decade, with the development of several process calculi with localities, including e.g. Mobile Ambients, Dπ, and Seal. We suggest to use the locality concept as a general form of component, that can be used, at the same time, as a unit of modularity, of isolation, of mobility, and of passivation. Specifically, we have designed Oz/K, a kernel programming language, that adds to the Oz computation model a notion of locality borrowed from the Kell calculus. We have defined an operational semantics for the language, and worked out several examples to illustrate how Oz/K supports open distributed programming.

6.3.2. A Type System for the Dream Framework

Building complex software architectures can be subject to subtle assemblage errors that are typically not captured by classical type systems available with host programming languages such as Java, C#, or ML. Our real-life motivating example is the Dream component-based framework (see 5.3), that has been developed to facilitate the construction of configurable communication subsystems. To avoid assemblage errors in the Dream framework, we introduce a new type system, together with its type checking and type inference algorithms. Our type system can actually be used with any model where components interact by exchanging structured messages on ports. Types in our system build approximations of component behaviors, and allow verifying that message handling operations remain consistent within a whole configuration. Technically, our type system combines extensible record types with row variables, together with set and process types. Inference is aided by the use of sparse type annotations, and operates by propagating type information in a configuration graph.

[3]D. Hirschkoff, T. Hirschowitz, D. Pous, A. Schmitt and J.-B. Stefani. Component-Oriented Programming with Sharing: Containment is not Ownership, *4th International Conference on Generative Programming and Component Engineering* (GPCE), Tallinn (Estonia) September 2005

6.3.3. *Bidirectional Programming Languages*

In collaboration with Benjamin C. Pierce at University of Pennsylvania, we have continued our exploration of bidirectional programming languages. In [16], we proposed a novel approach to the view update problem for tree-structured data: a domain-specific programming language in which all expressions denote bi-directional transformations on trees. In one direction, these transformations dubbed *lenses* map a “concrete” tree into a simplified “abstract view”; in the other, they map a modified abstract view, together with the original concrete tree, to a correspondingly modified concrete tree. Our design emphasizes both robustness and ease of use, guaranteeing strong well-behavedness and totality properties for well-typed lenses.

We identified a natural mathematical space of well-behaved bi-directional transformations over arbitrary structures, studied definedness and continuity in this setting, and stated a precise connection with the classical theory of “update translation under a constant complement” from databases. We then instantiated this semantic framework in the form of a collection of lens combinators that can be assembled to describe transformations on trees. These combinators include familiar constructs from functional programming (composition, mapping, projection, conditionals, recursion) together with some novel primitives for manipulating trees (splitting, pruning, copying, merging, etc.). We illustrated the expressiveness of these combinators by developing a number of bi-directional list-processing transformations as derived forms. An extended example showed how our combinators can be used to define a lens that translates between a native HTML representation of browser bookmarks and a generic abstract bookmark format.

6.4. Distributed Algorithms

Participants: Jean-Bernard Stefani, Vivien Quéma, Alan Schmitt.

This section describes new results on the design of distributed algorithms that are both sound and efficient. Most of this work was done in collaboration with groups at the University of Pennsylvania, École Polytechnique Fédérale de Lausanne and Università La Sapienza in Rome.

6.4.1. *Conflict Resolution and Optimistic Replication*

Current techniques for reconciling disconnected changes to optimistically replicated data often use version vectors or related mechanisms to track causal histories. This allows the system to tell whether the value at one replica dominates another or whether the two replicas are in conflict. However, current algorithms do not provide entirely satisfactory ways of repairing conflicts. The usual approach is to introduce fresh events into the causal history, even in situations where the causally independent values at the two replicas are actually equal. In some scenarios these events may later conflict with each other or with further updates, slowing or even preventing convergence of the whole system.

To address this issue, we enrich the set of possible actions at a replica to include a notion of explicit conflict resolution between existing events, where the user at a replica declares that one set of events dominates another, or that a set of events are equivalent. We precisely specify the behavior of this refined replication framework from a user’s point of view and show that, if communication is assumed to be “reciprocal” (with pairs of replicas exchanging information about their current states), then this specification can be implemented by an algorithm with the property that the information stored at any replica and the sizes of the messages sent between replicas are bounded by a polynomial function of the number of replicas in the system.

More information in [28].

6.4.2. *Eventual Consistency*

Many systems running on local area networks provide so-called single copy semantics that gives the user the illusion of accessing a single, highly available object. Typical solutions require users to access a quorum of replicas, to acquire exclusive locks on data they wish to update or to agree on a total order of updates to be applied at each replica. Maintaining single-copy semantics in a worldwide deployed system is practically very expensive and theoretically impossible. It is thus necessary to use weaker consistency criteria. This is precisely what eventual consistency provides. It guarantees that whatever the current state of the replica, if no new updates are issued and replicas can communicate freely for a long enough period, the contents of all replicas eventually become identical. From an implementation point of view, the issues to solve in order to guarantee eventual consistency are: (1) update dissemination: each update must eventually reach all replicas, and (2) update ordering: all updates must be eventually applied in the same order at each replica.

We have developed a replication protocol ensuring eventual consistency in large-scale distributed systems subject to network partitions and asynchrony. The protocol combines various self-stabilization techniques (partition merging, stable process election and gossip-based propagation). Moreover, it does not rely on any form of consensus, which would lead to block the replicas in case of partitions and asynchrony. Instead, the protocol ensures that (1) updates are continuously applied to the replicas and (2) no two updates are ever performed in a different order. Gaps might occur during periods of unreliable communication. They are filled whenever connectivity is provided, and consistency is then eventually ensured, but without any conscious commitment. That is, there is no point in the computation when replicas know that consistency is achieved. This unconsciousness is the key to tolerating perpetual asynchrony with no consensus support. A simulation study has shown that the our protocol is scalable and achieves high throughput under load.

More information in [20].

6.4.3. Total Order Broadcast

Uniform total order broadcast (UTOcast) is a fundamental communication primitive that plays a central role in bringing cheap software-based high vailability to a wide array of services. Our work consists in designing algorithms for implementing UTOcast in various environments (clusters, grid, real-time systems, etc.). Many studies have been done in order to decrease the message and time complexity (i.e. latency). Our work focuses on throughput. We aim at designing algorithms that provide higher throughput than currently available ones.

We have designed FSR, a uniform total order broadcast protocol dedicated to clusters of homogeneous machines. FSR provides high throughput, regardless of message broadcast patterns. FSR is based on a ring topology, only relies on point-to-point inter-process communication, and has a linear latency with respect to the number of processes. FSR is also fair in the sense that each process has an equal opportunity of having its messages delivered by all processes. FSR achieves a throughput of 79MBits/s on a cluster of Itanium based machines interconnected by a 100 Mbit/s switched Ethernet network.

More information in [29].

6.5. Flexible Operating Systems Kernels

Participants: Jean-Bernard Stefani, Renaud Lachaize, Ali Erdem Özcan, Juraj Polakovic.

Our work on flexible operating systems kernels is based on THINK [7], a component-based programming framework for building modular and customizable operating systems and applications. THINK can be considered as an implementation of the FRACTAL component model that enables component-based programming in the C language.

This work is described in the following sections.

6.5.1. Think-on-SoC

Multi-Processor Systems-on-Chips (MPSoCs), integrating multiple processors and hardware accelerators in a single chip, seem to be the successors of classical Application-Specific Integrated Circuits (ASIC) to meet programmability, performance and energy consumption requirements of mobile devices. Implementing complex multimedia applications and networking services on such highly-constrained embedded devices is a challenging task. THINK-on-SoC addresses this issue by implementing a lightweight component-based OS and application development framework, an extension of the existing THINK framework. This environment includes transparent remote component invocations and specific component abstractions for modeling the mapping of an OS or an application on a multi-processor platform.

We are working on programming models and tools to ease the development of streaming applications (e.g. multimedia decoders) on such parallel platforms. Recent work in this context has led to the development of ADL extensions, inspired by the Join-Calculus, for specifying the collaborative behavior of a set of components. Applications programmed according to this model can be transparently mapped to (and optimized for) various runtime environments.

THINK-on-SoC also explores some compile-time optimizations to improve memory and runtime performance by shrinking customized applications to fit the budget determined by the underlying MPSoC. For experimentations, we use a realistic MPSoC simulation platform provided by STMicroelectronics. The main results are described in [44], [40].

6.5.2. Dynamic Reconfiguration of Embedded Systems Kernels

We propose a programming model composed out of four different parts that can be combined to build a reconfigurable operating system - *i)* an ADL to build the software architecture of the system, *ii)* dynamic reconfiguration mechanisms and their specification, *iii)* a formalism to specify the execution model of the software architecture, *iv)* and a procedural reconfiguration domain specific language to program reconfigurations. Finally, in order to meet requirements of embedded systems, we explore how to optimize the component structures of such reconfigurable systems.

Dynamic reconfiguration involves the capture of the quiescent state of the target sub-system, the addition, removal and binding of relevant components and the state transfer from the old sub-system to the new one. In this context, recent work has focused on the development of ADL-based facilities supporting different mechanisms (namely MMLite and K42) for dynamic reconfiguration. In contrast to previous proposals, our approach allows the selection between these different mechanisms at build time, with little or no changes to operating system and application components [33].

In addition, in order to ease the development of safe and efficient reconfiguration protocols we use a domain-specific language (FScript) adapted to the THINK framework. We propose to offload the compilation process of reconfiguration programs to a non-constrained host. The such compiled program is then loaded and executed by the target system. This approach can be well-suited for capacity constrained embedded devices such as networked sensors.

The work on dynamically reconfigurable component-based operating systems kernels is the subject of the Ph.D. thesis of J. Polakovic (to be defended in 2007).

6.5.3. Multitarget Fractal

FRACTAL is a language independent component model, with current implementations in Java, C and C++. The goal of the Multitarget FRACTAL project is to regroup different implementations of FRACTAL within a single development environment. More precisely, our goal is to develop a unique ADL compiler that takes as input a unified ADL description and component implementations written in one of the above languages, and builds executable components for a specified platform (JVM for Java or a given hardware platform for C/C++). More generally, we are developing a unified and extensible toolset aimed at easing architecture-based software development. Our toolset supports heterogeneous architecture descriptions in various languages thanks to an extensible abstract syntactic tree (AST) architecture and a programmable component-based processing chain. This approach allows us to cope with a broad range of issues: support of a new input language, insertion of semantic analysis features, code generation for various programming languages and deployment of software components on various execution platforms [30].

The work on the construction of an extensible ADL toolset, as well as the design and the implementation of component-based OS and streaming applications for MPSoC (6.5.1), is the subject of the Ph.D. thesis of A. E. Özcan (to be defended in 2007).

6.6. Autonomous Distributed System Management

Our research on autonomous distributed system management is supported by an experimental platform, called JADE, whose design started in 2004. The infrastructure of JADE is a LAN-based cluster of Unix nodes.

The application domain is interactive, data-driven computing based on dynamic web content, an area that is representative of an important segment of distributed computing, for which availability, performance, and ease of run-time administration are crucial issues. We have selected J2EE as the initial middleware environment, since we have acquired experience in the design, use and evaluation of various components of this environment, specially in the context of OBJECTWEB (8.2).

Work on the JADE framework is decribed in 6.6.1. Specific results in several aspects of autonomic management are presented in 6.6.2 (configuration), 6.6.3 (performance and QoS), and 6.6.4 (recovery).

6.6.1. Jade: A Component-based Framework for Autonomic Computing

Participants: Jean-Bernard Stefani, Sara Bouchenak, Fabienne Boyer, Noël De Palma, Jakub Kornaś, Julien Legrand, Adrian Mos, Jérémy Philippe, Sylvain Sicard, Christophe Taton, Sacha Krakowiak.

The main thesis that we defend is that *architecture-based management is an adequate base for autonomic computing*. The design of the JADE framework is based on this principle. We therefore need (i) an explicit architectural description of both the managed and the managing elements; and (ii) explicit management interfaces for the various elements at all levels of description.

To that end, the current version of JADE uses the FRACTAL model, which answers both above requirements: the overall structure of the system is described by means of an ADL; and each component has explicit management interfaces allowing, among others, dynamic rebinding and reconfiguration. In order to integrate legacy systems (such as the tiers that make up a J2EE platform) in a FRACTAL environment, we use wrapping techniques that embed the management operations of the legacy system into FRACTAL controllers [31]. As an important side effect, the task of deploying such a “FRACTALized” application is greatly simplified (the size of the needed command scripts is reduced by an order of magnitude). See an example in 6.6.2.

[23] contains a summary of the design of JADE and a first evaluation.

Three aspects of autonomic management (configuration, performance management and recovery) using the JADE platform are described in the next sections. In 2006, we have initiated work on two additional aspects:

- *Self-protection*. The goal of self-protection is to maintain a system’s integrity in the face of attacks. To that end, we enable the protected system with a “sense of self”, i.e. an ability to detect the intrusion of foreign elements into the system. In a first phase, we have concentrated on the architectural level, by providing the system with the ability to detect an attempt to use the communication channels between the components in an illegal way. The reaction of the system to such an attempt is to isolate the nodes suspected to be compromised, by removing all bindings involving these nodes. Preliminary results are in [24], [25], [41], [26]. This work is the subject of the Ph.D. thesis of B. Claudel (started in 2006).
- *Large scale grid administration*. The goal here is to assist the deployment of large scale scientific applications on a grid, by extending the principles of JADE to an environment of thousands of machines. While these applications are usually statically deployed, we investigate dynamic adaptive deployment, in which the deployment process may dynamically evolve according to the requests and the available resources. Preliminary results are in [27], [42]. This work is the subject of the Ph.D. thesis of S. Fontaine (started in 2006).

6.6.2. Dynamic Configuration and Adaptation

Participants: Jean-Bernard Stefani, Takoua Abdellatif, Jakub Kornaś, Vivien Quéma.

Current Internet-based services require permanent availability and continuous evolution. These two demands may be conflicting if a server needs to be stopped for upgrade. Thus the issue of automated dynamic reconfiguration takes an increasing importance.

Updating a service at runtime involves three main tasks: (i) isolation of services as independent packages; (ii) deployment and redeployment of services; and (iii) handling service dependencies and state at runtime. We have investigated these three aspects, using J2EE servers as a testbed.

Points (i) and (ii) are tackled using FRACTAL as a base for implementing service packaging. Dependencies are expressed using FRACTAL bindings. The deployment infrastructure is also implemented using FRACTAL components, which allows it to be dynamically adaptable (e.g. by plugging various deployment policy modules). See [19].

To illustrate point (iii), we have reimplemented JONAS, an open source J2EE platform, by decomposing it in a hierarchy of parts, each of which is wrapped into a FRACTAL component [18]. Each component thus becomes an independent unit of configuration and deployment. The new system, called JONASALACARTE, then inherits all properties of a FRACTAL-based implementation, among which dynamic configurability. Experiments [17], [11] show that this is achieved with low overhead.

The goal of the ongoing research is to build a unified model and framework for the management tasks related to the packaging, deployment and reconfiguration process. We also need to consider such issues as versioning, as well as licensing issues (e.g. identifying independent units for the licensing aspect).

Methods and tools for component packaging and dynamic reconfiguration are the subject of the Ph.D. theses of T. Abdellatif [11] and J. Kornaś (ongoing).

6.6.3. Autonomous Management of Performance and QoS

Participants: Sara Bouchenak, Fabienne Boyer, Noël De Palma, Jérémy Philippe, Christophe Taton.

The goal of self-optimization is to maintain optimal (or near-optimal) system performance in spite of wide variations of the load or of the amount of available resources. Performance may be measured by various criteria, such as average response time or average throughput for an Internet service, or bounded jitter for a video server, etc.

In the first experiments, performed under the JADE framework (6.6.1), we have used an entire node as the resource allocation unit, and we have implemented a simple control loop based on thresholds. When the load (measured by CPU usage) goes over a preset threshold, a new node is allocated. When the load falls under another threshold, a node is released. Two instances of this control loop are used: one for the database tier, the other one for the application server tier. As a result, the system keeps the response time fairly stable under wide variations of the load. Preliminary results are reported in [23], [38], [39]. We are extending this work to other resources, and we examine the use of QoS related indices to define targets and evaluation criteria for autonomous resource management policies [32].

In the future, we plan to use methods and tools based on control theory. To that end, we have started a collaboration with NeCS, a new INRIA project working on control for networked systems.

This work is the subject of the Ph.D. theses of J. Philippe (QoS) and Ch. Taton (resource management), both started in 2005.

6.6.4. Autonomous Management of Recovery

Participants: Jean-Bernard Stefani, Sara Bouchenak, Fabienne Boyer, Noël De Palma, Sylvain Sicard.

The goal of self-repair is to restore the integrity of a system in the presence of failures. Up to now, we have considered a system running on a cluster of nodes, and recovery from node failures. Failures of a software component will be investigated in a further step.

This target system has the form of a set of interconnected components, equipped with wrappers according to the general scheme defined by the JADE framework (6.6.1). Critical components are replicated, and the consistency of the replicas is maintained during execution. In addition, the knowledge base maintained by the repair manager contains a representation of the system state, including the configuration of each component, the bindings between the components, and the placement of the components on the nodes of the cluster. This representation is causally connected to the controlled system and contains all the necessary information to reconstruct the system after a node failure.

The recovery process includes the following steps, according to the generic model of a control loop for autonomic computing.

- Detecting a failure, using a heartbeat technique.
- Planning the recovery sequence, i.e., allocating an available node; re-creating and re-configuring new instances of the failed components (those that were located on the failed node); restarting the restored components; updating the system representation in the knowledge base.
- Performing the actions defined above, using the services provided by the JADE framework.

More details are in [36]. This work is the subject of the Ph.D. thesis of S. Sicard, started in 2005.

7. Contracts and Grants with Industry

7.1. Collaboration with Bull

Participants: Jacques Mossière, Takoua Abdellatif.

The theme of the collaboration with Bull is the development of system software for exploiting clusters operating under Linux for scientific and data management applications. This collaboration started in October 2000, and also involves the Inria MESCAL project-team (Yves Denneulin).

Our contribution is the development of an administration support system for a cluster, with a specific emphasis on J2EE platforms.

In 2005 and 2006, we have experimented with a “fractalized” version of JONAS, an open source J2EE platform developed at Bull and distributed by the OBJECTWEB consortium. The resulting system (called JONASALACARTE) thus becomes easily extendable and adaptable, with low overhead. See 6.6.2 for more details and references.

This work started in September 2003 and was the subject of the Ph.D. thesis of T. Abdellatif [11] (Bull co-funding).

7.2. Collaboration with France-Telecom R&D

Participants: Jean-Bernard Stefani, Vivien Quéma, Jakub Kornaś.

SARDES maintains an active collaboration with France-Telecom R&D (Norbert Segard Center, Distributed Systems Architecture group):

1. Collaboration on the following aspects: extensions to the FRACTAL component model [13]; framework for dynamic reconfiguration of FRACTAL based applications (J.-B. Stefani, J. Kornaś, V. Quéma). Part of this work takes place within the OBJECTWEB consortium (8.2); see also 6.6.2.
2. Collaboration on distributed process calculi including mobility and distribution (6.2), and on the associated virtual machines (J.-B. Stefani).

7.3. Collaboration with STMicroElectronics

Participants: Jacques Mossière, Jean-Bernard Stefani, Ali Erdem Özcan.

The goal of this project, started in November 2003, is to investigate the use of the THINK framework to develop operating systems for on-chip multiprocessors. See 6.5.1 for a more detailed description.

This work is the subject of the Ph.D. thesis of A. E. Özcan (STMicroElectronics co-funding).

8. Other Grants and Activities

8.1. National Actions

8.1.1. ARP Network

SARDES is a member of the "Distributed Systems and Applications" group of the national research network on Systems Architecture, Networks and Parallelism (ARP: Architecture, Réseaux et Parallélisme).

See http://www.arp.cnrs.fr.

8.1.2. Project JOnES (ANR-RNTL)

Participants: Vivien Quéma, Pierre Garcia.

The goal of project JOnES is to develop an open source infrastructure for an Enterprise Service Bus, in the form of a set of functional components. The project is related to the ObjectWeb ESB initiative , which promotes an ESB "ecosystem" to federate various ESB open source offerings. The major innovation of JOnES is to propose a new distributed open framework for ESB, based on the interconnection of JBI-compliant containers. It is organized in three main tasks: ESB requirements and architecture, ESB internal framework, ESB services and technical components.

The project partners are INRIA (JACQUARD and SARDES projects, and ObjectWeb international consortium); EBM WebSourcing; ENSTIMAC (École des Mines d'Albi-Carmaux); France Telecom R&D, AMS lab; OpenWide; ScalAgent Distributed Technologies.

See http://www.rntl.org/projet/resume2005/jones.htm.

The project runs from January 2006 to December 2007.

8.1.3. Project SelfWare (ANR-RNTL)

Participants: Noël De Palma, Fabienne Boyer.

The goal of project Selfware is to provide a framework for autonomous management of large scale distributed applications including legacy software systems. The expected results are reducing operator mistakes (a major cause of failure for large Internet services), saving hardware ressources, and minimizing administrator efforts. The main challenge is to be able to manage any arbitrary complex legacy system with minimum effort. This framework will be evaluated for two actual middleware infrastructures: (i) a clustered J2EE application Server (JONAS) and (ii) a Message Oriented Middleware (JORAM). Experimental scenarios will illustrate self-optimizing and self-healing.

The project partners are INRIA Rhône Alpes; France Telecom R&D; École des Mines de Nantes; Bull; IRIT/ENSEEIHT, Toulouse.

The project runs from June 2006 to December 2008.

See http://www.rntl.org/projet/resume2005/selfware.htm

8.1.4. Project Modyfiable (ANR-ARA)

Participants: Alan Schmitt, Jean-Bernard Stefani.

Modern programming languages provide powerful abstractions for encouraging concise and modular programming, such as objects, classes, or functions. Nevertheless, most of these languages emphasize the algorithmic aspects of programs, as opposed to more pragmatic considerations such as dynamic linking and dynamic update, or distributed deployment. These aspects are forms of what we call "dynamic modularity": the modules may be manipulated during execution, as opposed to classical modularity, which only deals with modular program construction. Dynamic modularity is difficult to handle rigorously, especially for large programs. Component models such as .NET, CORBA, or EJB define methods and provide tools dedicated to dynamic modularity. Their success suggests the usefulness of a rigorous treatment of dynamic modularity.

From the technical standpoint, dynamic modularity covers many research fields in software engineering, such as dynamic linking and dynamic updating, concurrent and distributed processes, interoperability, modularity, multi-stage programming, and aspect-oriented programming. This apparent complexity makes it important to try and isolate the essence of dynamic modularity, with the aim of defining a kernel language dedicated to the specification and prototyping (or even compilation) of dynamically modular programs. This is the goal of MoDyFiable.

MoDyFiable consists of two research teams. The SARDES project (INRIA) has a long standing activity in component-based programming, including the development of a component model (FRACTAL), and operating systems based on this model. The PLUME team (École Normale Supérieure de Lyon) works, at a more theoretical level, on process algebras for mobile and distributed computing, and on the theory of programming languages, in particular regarding sophisticated forms of modularity.

Collaboration has started in 2006 between the two groups, with the definition of a first version of a process algebra for dynamic modularity. Ultimately, we plan to implement a prototype high-level programming language based on our process algebra. Programs written in this language (or at least in its process algebra core) should support formal reasoning using behavioral equivalences and specialized type systems.

The project runs from January 2006 to December 2008

8.1.5. Project AutoMan (ARC-INRIA)

Participant: Sara Bouchenak.

AutoMan is a Cooperative Research Initiative (ARC - Action de Recherche Coopérative) funded by INRIA. This project aims at devising a new breed of autonomic protocols to build highly scalable and available J2EE enterprise servers deployed and replicated on grid systems. Grid technology will provide the infrastructure to exploit a pool of available servers in order to achieve self-provisioning of resources. Autonomic management will continuously tune the system to maximize performance and availability with minimal human intervention.

The partners are INRIA (SARDES and OASIS project teams) and the Distributed Systems Laboratory, Technical University of Madrid, Spain

See http://sardes.inrialpes.fr/research/AutoMan

8.2. ObjectWeb Consortium

Participants: Jean-Bernard Stefani, Sacha Krakowiak.

OBJECTWEB is an open-source software community created at the end of 1999 by France Telecom R&D, Bull and Inria. Its goal is the development of open-source distributed middleware, in the form of flexible and adaptable components. These components range from specific software frameworks and protocols to integrated platforms. OBJECTWEB developments follow a systematic component-based approach.

In 2002, OBJECTWEB evolved into an international consortium hosted by Inria. The consortium is an independent non-profit organization open to companies, institutions and individuals.

SARDES contributes to OBJECTWEB through its technical involvement in the development of software components and frameworks (e.g. FRACTAL, C-JDBC, JOTM, DREAM, TRIBE) and through participation in the management structures of the consortium: board (J.-B. Stefani, past president and current member), and college of architects (J.-B. Stefani, current member).

See http://www.objectweb.org

8.3. Projects Funded by the European Commission

8.3.1. IST Project Gorda

Participants: Jean-Bernard Stefani, Sara Bouchenak.

Gorda (Open Replication of Databases) is an IST STREPS project, which aims at (i) promoting the interoperability of databases and replication protocols by defining generic architecture and interfaces that can be standardized; (ii) providing general purpose and widely-applicable database systems; and (iii) providing uniform techniques and tools for managing secure and heterogeneous replicated database systems. The project partners are Universidade do Minho (R. Oliveira), Università della Svizzera Italiana (F. Pedone), Universidade de Lisboa (L. Rodrigues), INRIA (SARDES project), Emic Networks (Finland) and MySQL AB (Sweden). The main contribution of SARDES is the C-JDBC technology, and the development of frameworks and tools for its use.

The project runs from October 2004 to September 2007.

See http://gorda.di.uminho.pt

8.3.2. IST Project SelfMan

Participants: Jean-Bernard Stefani, Alan Schmitt.

“Abnormal” events such as software updates, faults, threats, and performance hotspots become normal and even frequent occurrences in large distributed systems. The goal of SelfMan, an IST STREPS project, is to handle these events automatically by making the systems self managing: the systems will reconfigure themselves to handle changes in their environment or requirements without human intervention but according to high-level management policies. The focus is on four axes of self management, namely self configuration, self healing, self tuning, and self protection. The project partners are Université catholique de Louvain, Belgium; Royal Institute of Technology (Kungliga Tekniska Högskolan), Sweden; Institut National de Recherche en Informatique et Automatique (INRIA), France; France Telecom R& D, France; Konrad-Zuse-Zentrum für Informationstechnik Berlin, Germany; E-Plus Mobilfunk GmbH & Co. KG, Germany; and National University of Singapore, Singapore.

SARDES will contribute its expertise on self-managing systems based on a component framwork.

The project runs from June 1, 2006 to May 31, 2009

8.3.3. IST Project Grid4All

Participants: Noël De Palma, Nikos Parlavantzas.

Grid4All aims at enabling domestic users, non-profit organisations such as schools, and small enterprises, to share their resources and to access massive Grid resources when needed, envisioning a future in which access to resources is democratised, readily available, cooperative, and inexpensive. Examples include home users of an image editing application, school projects like volcanic eruption simulations, or small businesses doing data mining. Cooperation examples include joint homework between pupils, and international collaboration such as the edition of a multilingual newsletter between schools from different countries.

Grid4All goals entail a system pooling large amounts of cheap resources; a dynamic system satisfying spikes of demand; using self-management techniques to scale and adapt to changes in environment; supporting isolated, secure, dynamic, geographically distributed user groups and using secure peer-to-peer techniques to federate large numbers of small-scale resources into large-scale Grids. The technical issues addressed are aspects of security, support for multiple administrative and management authorities, self-management by combining the strong points of structured overlay P2P networks and that of component models, on-demand resource allocation, heterogeneity, and fault tolerance.

The partners of the project are France Telecom (FT R&D); Institut National de Recherche en Informatique et en Automatique (INRIA); The Royal Institute of Technology (KTH); Swedish Institute of Computer Science (SICS); Institute of Communication and Computer Systems (ICCS); University of Piraeus Research Center (UPRC); Universitat Politècnica de Catalunya (UPC); Rededia S.L. (REDEDIA);

The project runs from January 2006 to December 2008.

8.3.4. ITEA Project S4All

Participants: Jean-Bernard Stefani, Sébastien Jean.

S4All (Services for All) is a project funded by the ITEA program Information Technology for European Advancement). The project, launched in July 2005 for a duration of 2 years, is focused on the provision of a platform for easy creation of person-to-person communication and customized web services.The global aim of the project is to explore the technical solutions, and consolidate the existing ones that may appear suitable for building the following vision: A world of user-centric services that are easy to create, share and use. The platform is organized as a “service bus”, allowing flexible integration of existing applications, as well as new developments. The contribution of SARDES is expertise in the area of adaptive middleware and integration of applications.

The partners are six large companies: Alcatel CIT Research and Innovation (leader), Bull, Nokia, Schneider Electric, Thales, Vodafone; three SMEs: Capricode, mCentric, Xquark: six academic partners: Fraunhofer Fokus, Helsinki Institute for Information Technology, INRIA, INT, Univ. Joseph Fourier (IMAG), Univ. Politecnica de Madrid.

The project runs from July 2005 to June 2007.

See http://www.itea2.org/public/project_leaflets/S4ALL_profile_oct-05.pdf

8.4. International Networks and Working Groups

8.4.1. CoreGrid Network of Excellence (IST-004265)

SARDES is a member of CoreGrid, the European Research Network on Foundations, Software Infrastructures and Applications for large scale distributed, GRID and Peer-to-Peer Technologies, launched in 2004 for four years. Its objective is to coordinate European efforts in the area of Grid and Peer-to-Peer technologies. It gathers forty-two institutions. SARDES is contributing in the areas of programming models and software architecture (FRACTAL has been adopted as a basis for the component-based programming model of CoreGrid).

See http://www.coregrid.net/

8.5. International Bilateral Collaborations

8.5.1. Europe

SARDES maintains long term collaboration with several research groups in Europe:

- École Polytechnique Fédérale de Lausanne: Distributed Systems Laboratory (Prof. André Schiper), Distributed Programming Laboratory (Prof. Rachid Guerraoui), LabOS (Prof. Willy Zwaenepoel). Collaboration on fault tolerance and performance management for clustered applications. See joint papers [21], [29], [20]. Contact persons in SARDES: S. Bouchenak, V. Quéma.
- University of Lancaster, *Distributed Media Systems* (Prof. Gordon Blair), on adaptable middleware for multimedia communication. Contact person in SARDES: J.-B. Stefani.
- Trinity College, Dublin, *Distributed Systems Group* (Prof. Vinny Cahill), on distributed programming and clusters. A. Senart, a former SARDES Ph.D. student, has been staying with TCD since academic year October 2003. Contact persons in SARDES: N. De Palma, S. Krakowiak.
- University of Copenhagen, DIKU Laboratory *Distributed Systems Group* (Prof. Eric Jul, Dr. Jørgen Hansen), on system support for clustered servers (visits and shared software). Contact persons in SARDES: S. Krakowiak, R. Lachaize.
- Università di Roma, La Sapienza, Dipartimento di Informatica e Sistemistica (Prof. Roberto Baldoni). Collaboration on distributed algorithms for large scale systems (post-doctoral stay of Vivien Quéma, 2005-2006). See joint papers [14], [20] (also with EPFL). Contact person in SARDES: V. Quéma.
- Foundation for Research and Technology Hellas, Institute of Computer Science, CARV Laboratory (Prof. Angelos Bilas). Collaboration on flexible and autonomic storage systems (post-doctoral stay of Renaud Lachaize, 2005-2006). Contact person in SARDES: R. Lachaize.

8.5.2. U.S.A.

Collaboration is under way with the University of Pennsylvania, Philadelphia (Prof. Benjamin C. Pierce), on the Harmony universal synchronizer: distributed algorithms, programming language design, type systems (visits, shared software and joint publications [15], [16], [28]). Contact person in SARDES: A. Schmitt.

8.5.3. China

SARDES is actively pursuing contacts with several Chinese universites and R&D teams on middleware, both for research cooperation purposes and on behalf of the OBJECTWEB Consortium. For research cooperation, the most advanced contacts are with Peking University (Prof. Mei), on the subject of advanced architecture description languages and autonomic distributed system management, with the National University of Defense Technology (Prof. Wang), on autonomic distributed systems management, and with the South China University of Technology in Guangzhou (Prof. Xu), on J2EE systems management. Contact person in SARDES: J.-B. Stefani.

9. Dissemination

9.1. Community Service

J.-B. Stefani is a member of the editorial board of the journal *Annales des Télécommunications* and of the program committees of DAIS'06 and DOA'06. He also is a member of the scientific Advisory Board of NTT DoCoMO Labs, and a member of the Comité d'Évaluation of the French program RNTL (Research Network on Software Technologies).

A. Schmitt is a member of the program committees of PLAN-X 2007 and JFLA 2007. He is also in charge of a joint Imag-Inria seminar on "Distributed Systems and Data Management".

V. Quéma is a member of the program committees of the International Conference on Self-Organization and Autonomous Systems in Computing and Communications (SOAS'2006) and the International Conference on Software Engineering Advances (ICSEA'2006), and a reviewer for the Journal of Parallel and Distributed Computing (Elsevier).

S. Krakowiak is the chairman of the committee sponsored by *Specif* (the French Computer Science Researchers' and Teachers' Association) to select each year an outstanding Ph.D. thesis in Computer Science defended in France. He has done evaluation work for EVA, the evaluation agency of the Danish Ministry of Education, and OAQ, the Center of Accreditation and Quality Assurance of the Swiss Universities.

9.2. ICAR Summer School

The project has organized a summer school on the subject of "Middleware and Construction of Distributed Applications". This school, called ICAR'06 (*Intergiciel et Construction d'Applications Réparties*), took place at Autrans, near Grenoble, from August 28 to September 1, 2006, and gathered about 60 attendants, from both academia and industry. This school was the fifth in a series of similar events organized by the project (and its predecessors) in the last 10 years.

N. De Palma, S .Krakowiak and J. Mossière were the organizers of the school. N. De Palma, S. Krakowiak, J. Legrand, A. Schmitt, and Ch. Taton participated in the lectures and demos.

A book based on the contents of the school will be published in 2007.

See http://sardes.inrialpes.fr/ecole/2006/

9.3. University Teaching

S. Bouchenak, F. Boyer, N. De Palma, S. Krakowiak, R. Lachaize, J. Mossière, and V. Quéma have taught several operating systems and distributed systems courses at the M.S. and M.Eng. levels, both at Institut National Polytechnique de Grenoble and at université Joseph Fourier. Most of our Ph.D. students contributed to these courses as teaching assistants.

9.4. Participation in Seminars, Workshops, Conferences

Several members of SARDES attended various scientific conferences and workshops. See the relevant section of the Bibliography for details. Most of the publications of the project are available on line from the SARDES web site:

http://sardes.inrialpes.fr/

10. Bibliography

Major publications by the team in recent years

[1] S. KRAKOWIAK, S. SHRIVASTAVA (editors). *Recent Advances in Distributed Systems*, Lecture Notes in Computer Science 1752, Springer, 2000.

[2] G. BLAIR, J.-B. STEFANI. *Open Distributed Processing and Multimedia*, Addison-Wesley, 1997.

[3] S. BOUCHENAK, D. HAGIMONT, S. KRAKOWIAK, N. DE PALMA, F. BOYER. *Experiences Implementing Efficient Java Thread Serialization, Mobility and Persistence*, in "Software – Practice and Experience (SP&E)", vol. 34, n^{o} 4, april 2004, p. 355–394.

[4] É. BRUNETON, T. COUPAYE, M. LECLERCQ, V. QUÉMA, J.-B. STEFANI. *The Fractal Component Model and its Support in Java*, in "Software – Practice and Experience (SP&E)", special issue on “Experiences with Auto-adaptive and Reconfigurable Systems”, vol. 36, n^{o} 11-12, September 2006, p. 1257–1284.

[5] É. BRUNETON, M. RIVEILL. *An Architecture for Extensible Middleware Platforms*, in "Software – Practice and Experience (SP&E)", vol. 31, November 2001, p. 1237–1264.

[6] E. CECCHET, J. MARGUERITE, W. ZWAENEPOEL. *Performance and Scalability of EJB Applications*, in "Proceedings of OOPSLA'02, Seattle, WA, USA", November 2002.

[7] J.-PH. FASSINO, J.-B. STEFANI, J. LAWALL, G. MULLER. *THINK: A Software Framework for Component-based Operating System Kernels*, in "Proceedings of Usenix Annual Technical Conference, Monterey (USA)", June 10th-15th 2002.

[8] D. HAGIMONT, F. BOYER. *A Configurable RMI Mechanism for Sharing Distributed Java Objects*, in "IEEE Internet Computing", vol. 5, n^{o} 1, 2001, p. 36–44.

[9] É. NAJM, A. NIMOUR, J.-B. STEFANI. *Behavioural Typing for Objects and Process Calculi*, chap. 13, Cambridge University Press, 2001, p. 281–301.

[10] A. SCHMITT, J.-B. STEFANI. *The M-calculus: A Higher-Order Distributed Process Calculus*, in "In Proceedings of the 30th Annual ACM Symposium on Principles of Programming Languages (POPL'03), New Orleans, LA, USA", January 15-17 2003.

Year Publications

Doctoral dissertations and Habilitation theses

[11] T. ABDELLATIF. *Apport des architectures à composants pour l'administration des intergiciels*, Ph. D. Thesis, Institut National Polytechnique de Grenoble, September 2006, http://sardes.inrialpes.fr/papers/files/06-Abdellatif-PhD.pdf.

Articles in refereed journals and book chapters

[12] T. ABDELLATIF, A. DANES. *JMX-based autonomic management of J2EE servers*, in "International Transactions on Systems Science and Applications (ITSSA)", vol. 3/4, 2006.

[13] É. BRUNETON, T. COUPAYE, M. LECLERCQ, V. QUÉMA, J.-B. STEFANI. *The Fractal Component Model and its Support in Java*, in "Software – Practice and Experience (SP&E)", special issue on “Experiences with Auto-adaptive and Reconfigurable Systems”, vol. 36, n^{o} 11-12, September 2006, p. 1257–1284.

[14] G. CORTESE, F. MORABITO, F. DAVIDE, A. VIRGILLITO, R. BERALDI, V. QUÉMA. *Data Aggregation in Large-Scale Distributed Systems*, in "Emerging Communication: Studies in New Technologies and Practices in Communication", R. BALDONI, G. CORTESE, F. DAVIDE, A. MELPIGNANO (editors). , ISBN: 1-58603-629-7, IOS Press, July 2006, p. 53–78.

[15] J. N. FOSTER, M. B. GREENWALD, C. KIRKEGAARD, B. C. PIERCE, A. SCHMITT. *Exploiting Schemas in Data Synchronization*, in "Journal of Computer and System Sciences (JCSS), Special Issue on Database Theory", to appear, 2006.

[16] J. N. FOSTER, M. B. GREENWALD, J. T. MOORE, B. C. PIERCE, A. SCHMITT. *Combinators for Bi-Directional Tree Transformations: A Linguistic Approach to the View Update Problem*, in "ACM Transactions on Programming Languages and Systems (TOPLAS)", to appear, 2006.

Publications in Conferences and Workshops

[17] T. ABDELLATIF, F. BOYER, J. KORNAŚ, J.-B. STEFANI. *Administration fondée sur l'architecture des serveurs d'applications J2EE patrimoniaux*, in "5ème Conférence Française sur les Systèmes d'Exploitation (CFSE-5), Perpignan, France", October 2006.

[18] T. ABDELLATIF, A. DANES. *A simple approach to autonomic J2EE servers (short paper)*, in "IEEE International Conference on Self-Organization and Autonomic Systems in Computing and Communications (SOAS'2006), Erfurt, Germany", September 2006.

[19] T. ABDELLATIF, D. HOAREAU, Y. MAHÉO. *Automated deployment of enterprise systems in large-scale environments (short paper)*, in "International Symposium on Distributed Objects and Applications (DOA'2006), Montpellier, France", October 2006.

[20] R. BALDONI, R. GUERRAOUI, R. LEVY, V. QUÉMA, S. T. PIERGIOVANNI. *Unconscious Eventual Consistency with Gossips*, in "8th International Symposium on Stabilization, Safety, and Security of Distributed Systems (formerly Symposium on Self-stabilizing Systems) (SSS 2006), Dallas, TX, USA", Lecture Notes in Computer Science, vol. 4280, Springer, November 2006, p. 65–81.

[21] S. BOUCHENAK, A. COX, S. DROPSHO, S. MITTAL, W. ZWAENEPOEL. *Caching Dynamic Web Content: Designing and Analysing an Aspect-oriented Solution*, in "Middleware 2006, ACM/IFIP/USENIX 7th International Middleware Conference, Melbourne, Australia", November 2006.

[22] S. BOUCHENAK, N. DE PALMA, D. HAGIMONT, S. KRAKOWIAK, C. TATON. *Autonomic Management of Internet Services: Experience with Self-Optimization (short paper)*, in "3rd International Conference on Autonomic Computing (ICAC), Dublin, Ireland", June 2006, http://sardes.inrialpes.fr/papers/files/06-Bouchenak-ICAC.pdf.

[23] S. BOUCHENAK, N. DE PALMA, D. HAGIMONT, C. TATON. *Autonomic Management of Clustered Applications*, in "IEEE International Conference on Cluster Computing, Barcelona, Spain", September 2006, http://sardes.inrialpes.fr/papers/files/06-Bouchenak-Cluster.pdf.

[24] B. CLAUDEL, N. DE PALMA, R. LACHAIZE, S. BOUCHENAK, D. HAGIMONT. *Une approche architecturale pour l'auto-protection de systèmes répartis*, in "5ème Conférence Française sur les Systèmes d'Exploitation (CFSE 2006), Le Canet en Roussillon, France", October 2006, http://sardes.inrialpes.fr/papers/files/06-Claudel-CFSE.pdf.

[25] B. CLAUDEL, N. DE PALMA, R. LACHAIZE, D. HAGIMONT. *Self-protection for Distributed Component-Based Applications*, in "8th International Symposium on Stabilization, Safety, and Security of Distributed Systems (formerly Symposium on Self-stabilizing Systems) (SSS 2006), Dallas, TX, USA", November 2006, http://sardes.inrialpes.fr/papers/files/06-Claudel-SSS.pdf.

[26] N. DE PALMA, B. CLAUDEL, R. LACHAIZE, S. BOUCHENAK, D. HAGIMONT. *Self-Protected Systems: an experiment*, in "5th Conference on Security in Network Architectures (SAR), Seignosse, France", June 2006, http://sardes.inrialpes.fr/papers/files/06-Depalma-SAR.pdf.

[27] S. FONTAINE, C. TATON, S. BOUCHENAK, T. GAUTIER. *Administration autonome d'applications réparties sur grilles*, in "Rencontres francophones du Parallélisme (RenPar'17), Perpignan, France", October 2006, http://sardes.inrialpes.fr/papers/files/06-Fontaine-RenPar.pdf.

[28] M. B. GREENWALD, S. KHANNA, K. KUNAL, B. C. PIERCE, A. SCHMITT. *Agreeing to Agree: Conflict Resolution for Optimistically Replicated Data*, in "20th International Symposium on Distributed Computing (DISC), Stockholm, Sweden", Lecture Notes in Computer Science, vol. 4167, Springer, September 2006, p. 269–283, http://alan.petitepomme.net/papers/DISC_2006_nway.pdf.

[29] R. GUERRAOUI, R. LEVY, B. POCHON, V. QUÉMA. *High Throughput Uniform Total Order Broadcast Protocol for Cluster Environments*, in "International Conference on Dependable Systems and Networks (DSN), Philadelphia, PA, USA", IEEE Computer Society, June 2006, p. 549–557, http://sardes.inrialpes.fr/papers/files/06-Guerraoui-DSN.pdf.

[30] M. LECLERCQ, A. E. ÖZCAN, V. QUÉMA, J.-B. STEFANI. *Supporting Heterogeneous Architecture Descriptions in an Extensible Toolset*, to appear, May 2007.

[31] J. PHILIPPE, S. SICARD, C. TATON. *Component-Based Autonomic Management System (poster)*, in "5th Fractal Workshop, associated with ECOOP'2006, Nantes, France", July 2006, http://sardes.inrialpes.fr/papers/files/06-Philippe-FractalWS.pdf.

[32] J. PHILIPPE, N. DE PALMA, S. BOUCHENAK, F. BOYER, D. HAGIMONT. *A Black-box Approach for Web Application SLA (short paper)*, in "21st ACM Symposium on Applied Computing (SAC'06), Dijon, France", April 2006, http://sardes.inrialpes.fr/papers/files/06-Philippe-SAC.pdf.

[33] J. POLAKOVIC, A. E. ÖZCAN, J.-B. STEFANI. *Building Reconfigurable Component-Based OS with Think*, in "EUROMICRO '06: Proceedings of the 32nd EUROMICRO Conference on Software Engineering and Advanced Applications, Dubrovnik, Croatia", IEEE Computer Society, August 2006, p. 178–185, http://sardes.inrialpes.fr/papers/files/06-Polakovic-Dynamic.pdf.

[34] V. SCHIAVONI, V. QUÉMA. *A Posteriori Defensive Programming: an Annotation Toolkit for DoS-Resistant Component-Based Architectures*, in "21st ACM Symposium on Applied Computing (SAC'06), Dijon, France", ACM, April 2006, p. 1734–1738.

[35] A. SCHMITT, J.-B. STEFANI. *Towards a Calculus for Distributed Components*, in "5th International Symposium on Formal Methods for Components and Objects (FMCO 2006), Amsterdam, Netherlands", November 2006.

[36] S. SICARD, N. DE PALMA, D. HAGIMONT. *J2EE Server Scalability through EJB Replication*, in "21st ACM Symposium on Applied Computing (SAC'06), Dijon, France", April 2006, http://sardes.inrialpes.fr/papers/files/06-Sicard-SAC.pdf.

[37] J.-B. STEFANI. *Fractal: a component-based computation model for manageable systems (invited talk)*, in "HPC-GECO/CompFrame Joint Workshop, Paris, France", June 2006.

[38] C. TATON, S. BOUCHENAK, N. DE PALMA, D. HAGIMONT, S. KRAKOWIAK, J. ARNAUD. *Administration autonome de services Internet : Expérience avec l'auto-optimisation*, in "5ème Conférence Française sur les Systèmes d'Exploitation (CFSE 2006), Le Canet en Roussillon, France", October 2006, http://sardes.inrialpes.fr/papers/files/06-Taton-CFSE.pdf.

[39] C. TATON, S. BOUCHENAK, N. DE PALMA, D. HAGIMONT, S. SICARD. *Self-Optimization of Clustered Databases*, in "2nd International IEEE WoWMoM Workshop on Autonomic Communications and Computing (ACC 2006), Niagara-Falls, Buffalo-NY", June 2006, http://sardes.inrialpes.fr/papers/files/06-Taton-ACC.pdf.

[40] A. E. ÖZCAN, S. JEAN, J.-B. STEFANI. *Bringing Ease and Adaptability to MPSoC Software Design: A Component-Based Approach*, in "Construction and Analysis of Safe, Secure, and Interoperable Smart Devices, 2nd International Workshop, CASSIS'05, Nice, France", Lecture Notes in Computer Science, ISBN 3-540-33689-3, vol. 3956, Springer, 2006, p. 118–137, http://sardes.inrialpes.fr/papers/files/06-Ozcan-Cassis.pdf.

Miscellaneous

[41] B. CLAUDEL. *Auto-protection des systèmes autonomes – Application aux grappes J2EE*, Rapport de Master Recherche Informatique : Systèmes et Logiciel (Master's thesis), École Doctorale Mathématiques et Informatique, Grenoble, June 2006, http://sardes.inrialpes.fr/~claudel/publications/M2RReport.pdf.

[42] S. FONTAINE. *Systèmes autonomes – Application au calcul scientifique sur grilles*, Rapport de Master Recherche Informatique : Systèmes et Logiciel (Master's thesis), École Doctorale Mathématiques et Informatique, Grenoble, June 2006.

[43] S. LENGLET. *Bisimulations dans un π-calcul d'ordre supérieur avec passivation*, Rapport de Master Recherche Informatique (Master's thesis), École Normale Supérieure de Lyon, June 2006.

[44] F. MAYOT. *Programmation parallèle à base de composants pour les applications de streaming – Contribution à l'infrastructure Think*, Rapport de Master Recherche Informatique : Systèmes et Logi-

ciel (Master's thesis), École Doctorale Mathématiques et Informatique, Grenoble, September 2006, http://sardes.inrialpes.fr/papers/files/06-Mayot-M2R.pdf.

References in notes

[45] *Research Projects in Autonomic Computing*, 2003, http://www.research.ibm.com/autonomic/research/projects.html, IBM Research.

[46] *Proceedings of the 18th International Conference on Distributed Computing Systems*, IEEE Computer Society Press, Amsterdam, The Netherlands, May 1998.

[47] *Report of the IST Advisory Group concerning Software Techologies, Embedded Systems, and Distributed Systems: A European Strategy towards an Ambient Intelligent environment*, June 2002, http://www.cordis.lu/ist/istag.htm, Office for Official Publications of the European Communities.

[48] G. Blair, G. Coulson, A. Andersen, L. Blair, M. Clarke, F. Costa, H. Duran-Limon, T. Fitzpatrick, L. Johnston, R. Moreira, N. Parlavantzas, K. Saikoski. *The Design and Implementation of OpenORB v2*, in "IEEE Distributed Systems Online, vol. 2 no 6, Special Issue on Reflective Middleware", 2001.

[49] E. Dolstra, M. de Jonge, E. Visser. *Nix: A Safe and Policy-Free System for Software Deployment*, in "18th Large Installation System Administration Conference (LISA '04), Atlanta, Georgia, USA", L. Damon (editor). , USENIX, November 2004, p. 79-92, http://www.usenix.org/events/lisa04/.

[50] A. Flissi, P. Merle. *Une démarche dirigée par les modèles pour construire les machines de déploiement des intergiciels à composants*, in "L'Objet", Hermès, vol. 11, n^{o} 1-2, 2005, p. 79–94.

[51] J. O. Kephart, D. M. Chess. *The Vision of Autonomic Computing*, in "Computer - IEEE Computer Magazine", vol. 36, n^{o} 1, 2003, p. 41–50.

[52] J. O. Kephart. *Research Challenges of Autonomic Computing*, in "ICSE '05: Proceedings of the 27th international conference on Software engineering, New York, NY, USA", ACM Press, 2005, p. 15–22, http://doi.acm.org/10.1145/1062455.1062464.

[53] G. Kiczales, J. des Rivières, D. Bobrow. *The Art of the Metaobject Protocol*, MIT Press, 1991.

[54] G. Kiczales. *Aspect-Oriented Programming*, in "ACM Computing Surveys", vol. 28, n^{o} 4, 1996, 154, http://doi.acm.org/10.1145/242224.242420.

[55] 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, Amsterdam, The Netherlands", IEEE Computer Society Press, May 1998, p. 240–249.

[56] D. Oppenheimer, A. Ganapathi, D. A. Patterson. *Why do Internet services fail, and what can be done about it?*, in "4th USENIX Symposium on Internet Technologies and Systems (USITS '03)", March 2003.

[57] I. Piumarta, F. Riccardi. *Optimizing direct threaded code by selective inlining*, in "1998 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'98), Montreal, Canada", 1998.

[58] C. SZYPERSKI. *Component Software - Beyond Object-Oriented Programming*, 2nd ed., 589 pp., Addison-Wesley, 2002.