INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Sardes

System Architecture for Reflective Distributed Computing Environments

Grenoble - Rhône-Alpes

THEME COM

Activity Report

2007

Table of contents

1. Team

SARDES *is a project team of INRIA Grenoble-Rhône-Alpes and a research team of LIG (Grenoble Informatics Laboratory), a joint research unit (UMR 5217) of Centre National de la Recherche Scientifique (CNRS), Institut National Polytechnique de Grenoble (INPG) and université Joseph Fourier (UJF).*

Head of Project-team
Jean-Bernard Stefani [Research Director, Ingénieur Général des Télécommunications, HdR]

Administrative Assistant
Diane Courtiol

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]
Olivier Gruber [professor, université Joseph Fourier, HdR]
Sacha Krakowiak [professor emeritus, université Joseph Fourier, HdR]
Renaud Lachaize [associate professor, université Joseph Fourier]
Jacques Mossière [professor, Institut National Polytechnique de Grenoble, HdR]
Vivien Quéma [research scientist, Centre National de la Recherche Scientifique]

Post-doctoral Fellows
Marcia Pasin [University Santa Maria del Sud, Brazil, till Dec. 31, 2007]

Visitor
Didier Donsez [université Joseph Fourier, till Aug. 31, 2007]

Technical Staff
Julien Legrand [till Aug. 31, 2007]
Pierre Garcia [till Dec. 31, 2007]
Florent Métral
Nikos Parlavantzas
Valerio Schiavoni [from Feb. 15, 2007]
Alession Pace [from Feb. 15, 2007]
Lionel Debroux [from Oct. 1, 2007]
Fabien Motter [from Oct. 1, 2007]

Ph.D. Students
Jean Arnaud [government grant, from Oct. 1, 2007]
Benoît Claudel [government grant]
Stéphane Fontaine [government grant]
Fabien Gaud [government grant, from Oct. 1, 2007]
Jakub Kornaś [Inria grant (contract)]
Michaël Lienhardt [government grant]
Sergueï Lenglet [government grant]
Willy Malvaud [Inria grant (contract), from Oct. 1, 2007]
Juraj Polakovic [France Telecom RD]
Jérémy Philippe [Inria grant (contract)]
Sylvain Sicard [Inria grant (contract)]
Christophe Taton [government grant]

Interns
Ludovic Demontes [Master student, École Doctorale de Grenoble, from Oct. 1, 2007]
Sylvain Geneves [Master student, École Doctorale de Grenoble, from Oct. 1, 2007]

Sylvain Gonzales [Master student, CNAM, till Oct. 1, 2007]
Bhaskar Rauniyar [Master student, KTH, from Sep. 1, 2007]

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 OW2 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). It actively participates in the national research network on software technologies (RNTL), with two projets (SelfWare and JOnES) started in 2006, and two more (Flex-e-ware and ScorWare) started in 2007. It participates in the Minalogic initiative with one project (Aravis), started in Nov. 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.

A new permanent member has joined the project in Sep. 2007 (Olivier Gruber). A permanent member (Sébastien Jean) has left the project in Aug. 2007. Alan Schmitt has started a one year visit to the University of Bologna (Italy), to establish cooperation with Pr Davide Sangiorgi's team.

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 [38].

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 [52]. 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 [46] 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 [39], Flexinet, 2K/DynamicTAO) have been developed to investigate this approach.

The goal of aspect-oriented programming [47] (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 [48], code injection (inlining), dynamic code generation and optimization [50].

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 [5], 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 higher-order forms of components, bisimulation-based theory for higher-order calculi, dependent type systems for higher-order calculi with localities, support for dynamic reconfiguration, support for handling failures and recoverable activities, provably correct abstract machines for high-order component calculi, support for isolation and secure components.

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 main scientific challenges that we address, in this area, include: type systems and effective type checking for dynamic component structures, programming support for dynamic component structures, retargettable code generation for dynamic ADLs, 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 [49] 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 [44].

In the area of system administration, the role of configuration and deployment has long been neglected. Current research [41], [43], [37], [42] 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 [36] are active in this area. [45] 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, or to which SARDES heavily contributed, is available in the OW2 open source code base (see 8.2), which is accessible at http://www.ow2.org/.

5.2. Dream, a Framework for Building Asynchronous Middleware

Contact : see OW2 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 OW2 project distributed under an LGPL license.

See http://dream.objectweb.org

5.3. Jade, a Framework for Building Autonomic Management Systems

Contact : see OW2 page.

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 is available under a Cecill-C open source licence, as part of the JASMINe OW2 open source project. See http://forge.objectweb.org/projects/jasmine.

5.4. Fractal/Cecilia: Fractal component-based programming in C

Contact : see OW2 page.

FRACTAL is a component-based framework developed in cooperation by SARDES and FTR&D. Cecilia merges the original FRACTAL ADL compiler and the Think ADL compiler into an implementation of FRACTAL in C. Cecilia also extends the original FRACTAL ADL compiler into a retargettable compiler for heterogeneous architecture descriptions.

Cecila is available under an LGPL licence as part of the FRACTAL OW2 open source project. See http://forge.objectweb.org/projects/fractal.

5.5. C-Jdbc: Clustered JDBC-accessed Database

Contact : see OW2 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 OW2 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

5.6. Benchmarks for J2EE Systems

Contact : see OW2 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 OW2 JMOB (Java Middleware Open Benchmarking) project and are distributed under an LGPL license.

See http://jmob.objectweb.org

5.7. Jotm: Java Open Transactional Manager

Contact : see OW2 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 OW2 project distributed under an LGPL license.

See http://jotm.objectweb.org

[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.

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, Serguei Lenglet, Michaël Lienhardt.

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 basis for FRACTAL.

The M-calculus [9] 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 [10] 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.

Work on a bisimulation theory for the Kell calculus has continued in 2007. The problem of characterizing the contextual equivalence between Kell calculus programmes by a tractable bisimulation equivalence is still open, but we have made progress by studying a simpler calculus, the higher-order π-calculus with passivation (HOπP). We have found a normal bisimulation for HOπP without restriction, and shown that notions of normal bisimulation directly inspired by Sangiorgi's work on the higher-order π-calculus [51], do not work for HOπP with restriction [30].

6.3. Programming Languages

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

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 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. A first version of Oz/K has been published [26], [31], with proofs of basic security properties.

6.3.2. A Type System for the Dream Framework

Building complex component-based software architectures can lead to assemblage errors that are not captured by classical type sys- tems of host programming languages such as Java, C++, or ML. These assemblage errors can occur in particular when using component-based frameworks for building communication subsystems and middleware, for instance when using incompatible protocol stacks in a client and a server. A case in point is the DREAM framework, developed by SARDES. To address these assemblage issues, we have defined [32] a domain specific type system that can be used to avoid assemblage errors in DREAM assemblage. Our approach relies on the definition of a small process calculus that captures the operational essence of the Dream framework, and on the definition of a novel type system that combines rows à la Rémy with a form of process types à la Yoshida. Type infer- ence is undecidable for this type system, but we have defined an algorithm for semi-inference which makes our approach usable in practice.

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, a novel approach to the view-update problem for tree-structured data. We have defined a domain-specific programming language in which all expressions denote bidirectional transformations on trees [18]. In one direction, these transformations 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. The language relies on a collection of so-called lens-combinators that can be assembled to describe bidirectional 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, merging, etc.).

6.3.4. Static analysis of XML paths and types

In collaboration with the WAM project team at INRIA Grenoble-Rhône-Alpes, we have developed an algorithm to solve XPath decision problems under regular tree type constraints, and show its use to statically type-check XPath queries [23]. To this end, we have proved the decidability of a logic with converse for finite ordered trees whose time complexity is a simple exponential of the size of a formula. The logic corresponds to the alternation free modal ?-calculus without greatest fixpoint, restricted to finite trees, and where formulas are cycle-free. Our proof method is based on two auxiliary results. First, XML regular tree types and XPath expressions have a linear translation to cycle-free formulas. Second, the least and greatest fixpoints are equivalent for finite trees, hence the logic is closed under negation. Building on these results, we have described a practical, effective system for solving the satisfiability of a formula. The system has been experimented with some decision problems such as XPath emptiness, containment, overlap, and coverage, with or without type constraints. The benefit of the approach is that our system can be effectively used in static analyzers for programming languages manipulating both XPath expressions and XML type annotations (as input and output types).

These results are also interesting for the formalization of the Fscript language [40], developed by the Obasco team at INRIA Bretagne in Nantes, which allows to query and navigate FRACTAL component assemblages, and can be used as a basis of a domain-specific component configuration language (see 6.5.2).

6.4. Distributed Algorithms

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

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

6.4.1. Distributed event-based systems

The completely decoupled interaction model offered by the publish/subscribe communication paradigm perfectly suits the interoperability needs of todays large-scale, dynamic, peer-to-peer applications. The dynamic environments, where these applications are expected to work, pose a series of problems (potentially wide number of partipants, low-reliability of nodes, absence of a centralized authority, etc.) that severely limit the scalability of existing approaches which were originally thought for supporting distributed applications built on the top of static and managed environments. In collaboration with R. Baldoni's team at the University of Roma La Sapienza, we have proposed an architecture for implementing the topic-based publish/subscribe paradigm in large scale peer-to-peer systems, which is described in [17]. The architecture is based on clustering peers subscribed to the same topic. The major novelty of this architecture lies in the mechanism employed to bring events from the publisher to the cluster (namely outer-cluster routing). The evaluation shows that this mechanism for outer-cluster routing has a probability to bring events to the destination cluster very close to 1 while keeping small the involved number of out-of-cluster peers. Finally, the overall architecture is shown to be scalable along several fundamental dimensions like number of participants, subscriptions, and to exhibit a fair load distribution (load distribution closely follows the distribution of subscriptions on nodes).

6.4.2. Atomic storage

Following our previous work on efficient total order broadcast in clusters of machines, we have developed, in cooperation with R. Guerraoui's team at EPFL, an algorithm to ensure the atomicity of a distributed storage that can be read and written by any number of clients. The algorithm is presented in the paper [25]. In failure-free and synchronous situations, and even if there is contention, our algorithm has a high write throughput, and a read throughput that grows linearly with the number of available servers. The algorithm is devised with a homogeneous cluster of servers in mind. It organizes servers around a ring and assumes point-to-point communication. It is resilient to the crash failure of any number of readers and writers as well as to the crash failure of all but one server. We evaluated our algorithm on a cluster of 24 nodes with dual fast ethernet network interfaces (100 Mbps). We achieve 81 Mbps of write throughput and 8?90 Mbps of read throughput (with up to 8 servers) which conveys the linear scalability with the number of servers.

6.5. Flexible Operating Systems Kernels

Participants: Jean-Bernard Stefani, Renaud Lachaize, Ali Erdem Ozcan, Juraj Polakovic, Vivien Quema, Fabien Gaud.

Our work on flexible operating systems kernels is based on THINK [6], 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 kernels and applications on MPSoCs. We study in particular the use of event-based programming models, which promise to be both more convenient to program with, more amenable for the development of applications with throughput and latency constraints, and more scalable in high throughput contexts. 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 A.E. Ozcan's PhD thesis [11], and in F. Gaud's Master thesis [34].

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.

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 early 2008), and is presented in the papers [27], [14].

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 work was to regroup different implementations of FRACTAL within a single development environment. More precisely, our goal was 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. The work on the construction of the first version of an extensible ADL toolset, is documented in A.E. Ozcan's PhD thesis [11], and in the paper [7].

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. 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 OW2 (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, Nikos Parlavantzas, 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. As an important side effect, the task of deploying such a “FRACTAL-ized” 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.

[2], [4] and [13] present the design of JADE.

Three aspects of autonomic management (configuration, performance management and recovery) using the JADE framework are described in the next sections.

6.6.2. Dynamic Configuration and Adaptation

Participants: Jean-Bernard Stefani, Jakub Kornaś, Christophe Taton.

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.

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, taking into account issues such issues as versioning, and licensing (e.g. identifying independent units for the licensing aspect).

The application of our current results to a J2EE environment are documented in the paper [12], and will be further detailed in the forthcoming thesis of J. Kornas.

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.

After initial results with multi-tier J2EE application servers, we have studied the application of the JADE framework to the self-optimization of message-oriented middleware (MoM).

In the Java world, a standardized interface exists for MOMs: Java Messaging Service or JMS. JMS implementations typically use clustering techniques to provide some level of performance and fault-tolerance. One such implementation is JORAM, which is open-source and hosted by the OW2 consortium. We have modeled the performance various clustering configurations for JORMA, and validated our model with performance evaluation in a real-life cluster. In doing that, we observed that the resource-efficiency of the clustering methods can be very poor due to local instabilities and/or global load variations. To solve these issues, we have used the JADE framework to build autonomic capabilities on top of the JORAM middleware, and described how to (i) dynamically adapt the load distribution among the servers (load-balancing aspect) and (ii) dynamically adapt the replication level (provisioning aspect). These results are reported in [29], [15].

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, e.g. using a heartbeat technique.
- Planning the recovery sequence, e.g., 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.

Software engineering aspects of autonomous recovery are discussed in [28]. 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 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):

Collaboration concerns the following aspects:

- Further development of the FRACTAL component model.
- The development of a framework for dynamic reconfiguration of FRACTAL based applications, see 6.6.2.
- The development of the JADE framework, and its application to large scale distributed environments, see 8.3.2,8.3.3.

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 (Ecole 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; Ecole 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)

Participants: Sara Bouchenak, Marcia Pasin, Stéphane Fontaine.

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

8.1.6. Project ScorWare (ANR-RNTL)

Participants: Vivien Quéma, Renaud Lachaize, Valerio Schiavoni.

The SCOrWare project aims at providing an open source implementation of the recent Service Component Architecture (SCA) specifications defined by the Open SOA collaboration, an industrial consortium in the domain of software engineering, including BEA Systems, IBM Corporation, IONA Technologies, Oracle, Red Hat, Rogue Wave Software, Siemens, Sun Microsystems, and Sybase. Briefly, SCA defines a new architecture and programming model for Service Oriented Applications (SOA) based on the component paradigm and supporting several service description languages like WSDL and Java interfaces, several programming languages such as Java, C++, and BPEL, several communication protocols between applications such as SOAP, CORBA, Java RMI, and JMS.

The project aims to build a configurable and manageable SCA platform built upon the Fractal component model, with advanced functionalities such as dynamic reconfiguration of SCA component assemblies, a binding factory supporting different communication protocols between SCA components, a transaction service, a semantic trading service of SCA components, a deployment engine of autonomous SCA architecture and graphical administration consoles. The project partners are INRIA; Artenum; EBM WebSourcing; Edifixio; INT, Evry; IRIT, Toulouse; Obeo; OpenWide.

The project runs from January 2007 to December 2008.

See http://www.scorware.org

8.1.7. Project Flex-e-Ware (ANR-RNTL)

Participants: Vivien Quéma, Renaud Lachaize, Alession Pace.

The Flex-e-Ware project aims at defining software development tools and mechanisms dedicated to flexible, reconfigurable embedded systems. It aims to develop a common platform for the development of component-based embedded systems, based on the integration of the FRACTAL and Lightweight CCM component models.

The project partners are INRIA; CEA List; ENST, Paris; LIP6, Paris; France Telecom; Schneider Electric; STMicroelectronics; Teamlog; Trialog; Thales.

The project runs from January 2007 to December 2009.

See http://www.flex-eware.org/

8.2. OW2 Consortium

Participants: Jean-Bernard Stefani, Sacha Krakowiak.

OW2 is an open-source software consortium that has been created as the successor of the ObjectWeb consortium put in place in 2002 by Bull, France Telecom and INRIA. Its goal is the development of open-source distributed middleware.

SARDES contributes to OW2 through its technical involvement in the development of software components and frameworks (e.g. FRACTAL, DREAM, THINK, JADE) and through participation in the management structures of the consortium (J.-B. Stefani, member of the Technical Council).

See http://www.ow2.org/

8.3. Projects Funded by the European Commission

8.3.1. IST Project Gorda

Participants: Sara Bouchenak, Christophe Taton.

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, Vivien Quema.

"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: Jean-Bernard Stefani, 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.

8.4. 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 collaboration with several research groups in Europe:

- École Polytechnique Fédérale de Lausanne: Distributed Programming Laboratory (Prof. Rachid Guerraoui), LabOS (Prof. Willy Zwaenepoel). Collaboration on fault tolerance and performance management for clustered applications. Contact persons in SARDES: S. Bouchenak, V. Quéma.
- 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.
- 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). 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.
- Université Catholique de Louvain (Belgium): *Department of Computing Science and Engineering* (Prof. Peter Van Roy). Collaboration on component-based distributed programming, autonomic systems. Contact person in SARDES: J.B. Stefani.
- Swedish Institute of Computer Science (SICS, Stockholm): *Computer System Laboratory* (Prof. Seif Haridi). Collaboration on large scale distributed systems management, component-based distributed programming (9-month visit of Noel De Palma in 2008). Contact persons in SARDES: J.B. Stefani, N. De Palma.
- Università di Bologna (Italy): *Department of Computer Science* (Prof. Davide Sangiorgi). Collaboration on distributed process calculi, type systems for components (one-year visit of Alan Schmitt, 2007-2008). Contact persons in SARDES: A. Schmitt, J.B. Stefani.

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 and bi-directional languages: distributed algorithms, programming language design, type systems (visits, shared software and joint publications). 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 OW2 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 Téécommunications* and of the program committees of DAIS'07, DOA'07, FORTE '07, HPC-GECO '07 and Middleware '07. He as also been a member of the scientific Advisory Board of NTT DoCoMO European Labs (till Aug. 2007). He is a member of the OW2 Technical Council.

A. Schmitt is a member of the program committees of PLAN-X 2007 and JFLA 2007.

9.2. 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.

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

[2] S. BOUCHENAK, F. BOYER, S. KRAKOWIAK, D. HAGIMONT, A. MOS, N. DE PALMA, V. QUÉMA, J. STEFANI. *Architecture-Based Autonomous Repair Management: An Application to J2EE Clusters*, in "24th IEEE Symposium on Reliable Distributed Systems (SRDS 2005)", IEEE Computer Society, 2005.

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

[5] 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)", vol. 36, n^{o} 11-12, 2006.

[6] 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.

[7] M. LECLERCQ, A. E. ÖZCAN, V. QUÉMA, J.-B. STEFANI. *Supporting Heterogeneous Architecture Descriptions in an Extensible Toolset*, in "29th International Conference on Software Engineering (ICSE)", IEEE Computer Society, 2007.

[8] É. NAJM, A. NIMOUR, J.-B. STEFANI. *Behavioural Typing for Objects and Process Calculi*, in "Formal Methods for Distributed Processing", H. BOWMAN, J. DERRICK (editors), Cambridge University Press, 2001.

[9] 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)", 2003.

[10] A. SCHMITT, J.-B. STEFANI. *The Kell Calculus: A Family of Higher-Order Distributed Process Calculi*, in "Global Computing", Lecture Notes in Computer Science, vol. 3267, Springer, 2005.

Year Publications

Doctoral dissertations and Habilitation theses

[11] A. E. ÖZCAN. *Conception et Implantation d'un Environnement de Développement de Logiciels à Base de Composants, Applications aux Systèmes Multiprocesseurs sur Puce*, Ph. D. Thesis, Institut National Polytechnique de Grenoble, 2007, http://sardes.inrialpes.fr/papers/files/07-Ozcan-PhD.pdf.

Articles in refereed journals and book chapters

[12] T. ABDELLATIF, J. KORNAS, J.-B. STEFANI. *Reengineering J2EE Servers for Automated Management in Distributed Environments*, in "IEEE Distributed Systems Online", vol. 8, n^{o} 11, 2007.

[13] N. DE PALMA, S. BOUCHENAK, F. BOYER, D. HAGIMONT, S. SICARD, C. TATON. *Jade : Un Environnementd'Administration Autonome*, in "Techniques et Sciences Informatiques", to appear, 2008.

[14] J. POLAKOVIC, J.-B. STEFANI. *Architecting Reconfigurable Component-Based Operating Systems*, in "Journal of Systems Architecture", to appear, 2008.

[15] C. TATON, N. DE PALMA, J. PHILIPPE, S. BOUCHENAK. *Self-Optimization of Clustered Message-Oriented Middleware*, in "International Journal of Autonomic Computing", to appear, 2008.

Publications in Conferences and Workshops

[16] J. ARNAUD, S. BOUCHENAK. *Gestion de Ressources dans les Services Internet*, in "6e Conférence Française des Systèmes d'Exploitation (CFSE-6)", to appear, 2008.

[17] R. BALDONI, R. BERALDI, V. QUÉMA, L. QUERZONI, S. T. PIERGIOVANNI. *TERA: topic-based event routing for peer-to-peer architectures*, in "1st International Conference on Distributed Event-Based Systems", ACM, 2007.

[18] A. BOHANNON, J. N. FOSTER, B. C. PIERCE, A. PILKIEWICZ, A. SCHMITT. *Boomerang: resourceful lenses for string data*, in "35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, (POPL)", To appear, 2008.

[19] P. BRAND, J. HOGLUND, K. POPOV, N. DE PALMA, F. BOYER, N. PARLAVANTZAS, V. VLASSOV, A. AL-SHISHTAWY. *The Role of Overlay Services in a Self-Managing Framework for Dynamic Virtual Organisations*, in "CoreGRID Workshop on Grid Programming Model, Grid and P2P Systems Architecture and Grid Systems, Tools and Environments", 2007.

[20] T. COUPAYE, J.-B. STEFANI. *Fractal Component-Based Software Engineering – Report of Fractal CBSE Workshop at ECOOP'06*, in "ECOOP 2006 Workshop Reader", Lecture Notes in Computer Science, vol. 4379, Springer, 2007.

[21] N. DE PALMA, ET AL. *Spécification de Politiques d'AdministrationAutonome avec Tune*, in "6e Conférence Française des Systèmes d'Exploitation (CFSE-6)", to appear, 2008.

[22] J. N. FOSTER, B. C. PIERCE, A. SCHMITT. *A Logic Your Typechecker Can Count On: Unordered Tree Types in Practice*, in "ACM SIGPLAN Workshop Programming Language Technologies for XML (PLAN-X)", 2007.

[23] P. GENEVÈS, N. LAYAIDA, A. SCHMITT. *Efficient Static Analysis of XML Paths and Types*, in "PLDI '07: Proceedings of the 2007 ACM SIGPLAN conference on Programming language design and implementation", ACM Press, 2007.

[24] P. GENEVÈS, N. LAYAIDA, A. SCHMITT. *XPath Typing Using a Modal Logic with Converse for Finite Trees*, in "ACM SIGPLAN Workshop Programming Language Technologies for XML (PLAN-X)", 2007.

[25] R. GUERRAOUI, D. KOSTIC, R. R. LEVY, V. QUÉMA. *A High Throughput Atomic Storage Algorithm*, in "27th IEEE International Conference on Distributed Computing Systems (ICDCS 2007)", IEEE Computer Society, 2007.

[26] M. LIENHARDT, A. SCHMITT, J.-B. STEFANI. *Oz/K: A Kernel Language for Component-Based Open Programming*, in "6th ACM International Conference on Generative Programming and Component Engineering (GPCE)", ACM Press, 2007.

[27] J. POLAKOVIC, S. MAZARE, J.-B. STEFANI, P.-C. DAVID. *Experience with safe dynamic reconfigurations in component-based embedded systems*, in "10th ACM SIGSOFT Symposium on Component-Based Software Engineering (CBSE)", Lecture Notes in Computer Science, vol. 4608, Springer, 2007.

[28] S. SICARD, F. BOYER, N. DE PALMA. *Using Components for Architecture-Based Management: The Self-Repair Case*, in "30th International Conference on Software Engineering (ICSE)", to appear, 2008.

[29] C. TATON, N. DE PALMA, D. HAGIMONT, S. BOUCHENAK, J. PHILIPPE. *Self-optimization of Clustered Message-Oriented Middleware*, in "Distributed Objects and Applications (DOA) 2007 International Conference", Lecture Notes in Computer Science, vol. 4803, Springer, 2007.

Internal Reports

[30] S. LENGLET, A. SCHMITT, J.-B. STEFANI. *Bisimulations in Calculi Featuring Passivation and Restriction*, to appear, Technical report, Institut National de Recherche en Informatique et Automatique (INRIA), France, 2008.

[31] M. LIENHARDT, A. SCHMITT, J.-B. STEFANI. *Oz/K: A Kernel Language for Component-Based Open Programming*, Technical report, n^{o} RR-6202, Institut National de Recherche en Informatique et Automatique (INRIA), France, 2007, http://hal.inria.fr/inria-00149612.

[32] M. LIENHARDT, A. SCHMITT, J.-B. STEFANI. *Typing Communicating Component Assemblages*, to appear, Technical report, Institut National de Recherche en Informatique et Automatique (INRIA), France, 2008.

Miscellaneous

[33] J. ARNAUD. *Gestion de ressources dans les services Internet multi-niveaux – Modélisation et optimisation*, Technical report, Université Joseph Fourier, Grenoble, France, 2007, http://sardes.inrialpes.fr/papers/files/07-Arnaud-MasterThesis.pdf.

[34] F. GAUD. *Gestion autonome de flots d'exécution événementiels*, Technical report, Université Joseph Fourier, Grenoble, France, 2007, http://sardes.inrialpes.fr/papers/files/07-Gaud-MasterThesis.pdf.

[35] W. MALVAULT. *Protocoles de diffusion totalement ordonnée sur systèmes distribués synchrones*, Technical report, Université Joseph Fourier, Grenoble, France, 2007, http://sardes.inrialpes.fr/papers/files/07-Malvault-MasterThesis.pdf.

References in notes

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

[37] A. DEARLE, S. EISENBACH (editors). *Component Deployment, 3rd International Working Conference, CD 2005*, Lecture Notes in Computer Science, vol. 3798, Springer, 2005.

[38] *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.

[39] 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.

[40] P. DAVID. *Développement de composants Fractal adaptatifs: un langage dédié à l'aspects d'adaptation*, Ph. D. Thesis, Université de Nantes, France, 2005.

[41] 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/.

[42] S. FAGORZI, E. ZUCCA. *A Calculus of Components with Dynamic Type-Checking*, in "Electr. Notes Theor. Comput. Sci.", vol. 182, 2007.

[43] 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.

[44] 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.

[45] 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.

[46] G. KICZALES, J. DES RIVIÈRES, D. BOBROW. *The Art of the Metaobject Protocol*, MIT Press, 1991.

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

[48] 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.

[49] 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.

[50] 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.

[51] D. SANGIORGI. *Bisimulation for higher-order process calculi*, in "Information and Computation,Vol. 131, No 2", 1996.

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