INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

# *Project-Team ARLES*

# *Software Architectures and Distributed Systems*

*Paris - Rocquencourt*

THEME COM

# Table of contents

# 1. Team

**Research Scientist**

Valérie Issarny [ Team Leader, INRIA Research Director, HdR ]
Nikolaos Georgantas [ INRIA Research Scientist ]

**Faculty Member**

Pascal Poizat [ Assistant Professor, on leave from University of Evry, France, until August 2008 ]

**Technical Staff**

Amel Bennaceur [ *Ingénieure Associée*, INRIA ]
Sébastien Bianco [ *Ingénieur Associé*, INRIA, until October 2008 ]
Mauro Caporuscio [ *Ingénieur Expert*, INRIA EXOTICUS ]
Hassine Moungla [ *Ingénieur Expert*, INRIA PLASTIC, until July 2008 ]
Pierre-Guillaume Raverdy [ *Ingénieur Expert*, INRIA PLASTIC, until March 2008 ]
Pushpendra Singh [ *Ingénieur Expert*, INRIA EXOTICUS ]
Graham Thomson [ *Ingénieur Expert*, INRIA Amigo, until February 2008 ]

**PhD Student**

Nebil Ben Mabrouk [ INRIA CORDI fellowship, University Pierre and Marie Curie - Paris 6 ]
Sandrine Beauche [ DGA fellowship, University Pierre et Marie Curie - Paris 6 ]
Vasileios Fotopoulos [ INRIA CORDI fellowship, University Pierre et Marie Curie - Paris 6, since October 2008 ]
Manel Fredj [ INRIA ftellowship, University Pierre et Marie Curie - Paris 6 ]
Roberto Speicys Cardoso [ INRIA fellowship, University Pierre et Marie Curie - Paris 6 ]

**Post-Doctoral Fellow**

Fabio Mancinelli [ INRIA, until November 2008 ]
Animesh Pathak [ INRIA, since July 2008 ]

**Visiting Scientist**

Oleg Davidyuk [ PhD Student, University of Oulu, Finland, since September 2008 ]
Natallia Kokash [ PhD Student, University of Trento, Italy, until January 2008 ]
Romina Spalazzese [ PhD Student, University of L'Aquila, Italy, October 2008 – November 2008 ]
Apostolos Zarras [ Lecturer, University of Ioannina, Greece, July 2008 – August 2008 ]

**Administrative Assistant**

Emmanuelle Grousset [ until March 2008 ]
Nadia Mesrar [ April 2008 – August 2008 ]
Sylvie Loubressac [ since September 2008 ]

# 2. Overall Objectives

## 2.1. Overall Objectives

The main purpose of middleware is to overcome the heterogeneity of the distributed infrastructure. Middleware establishes a software layer that homogenizes the infrastructure diversities by means of a well-defined and structured distributed programming model. Moreover, middleware provides building blocks to be exploited by applications for enforcing non-functional properties, such as dependability and performance. Finally, by providing reusable solutions to the development of distributed systems, which is increasingly complex, the role of middleware has proved central in the software system development practice.

However, the development of distributed systems using middleware remains a complex task[1]. In particular, middleware must define the adequate networking and computing abstractions to match the distributed application requirements. Further, while the development of legacy middleware has been significantly driven by requirements of distributed information systems, the ongoing evolution of the networking environment leads to a much broader application of distributed computing. As a result, new requirements arise for middleware, e.g., supporting open and mobile networking, and context awareness [6].

In the above context, ARLES studies the engineering of middleware-based systems, with a special emphasis on enabling the ubiquitous computing[2] and subsequent pervasive computing[3] and ambient intelligence[4] visions. Our research is more specifically focused on eliciting middleware for pervasive computing, from foundations and architectural design to prototype implementations. Proposed middleware shall then effectively leverage networked resources, in particular accounting for advanced wireless networking technologies. This raises a number of complementary research challenges:

- **System architecture:** How to architect and further program pervasive computing systems out of the resources available in the highly dynamic networking environment?
- **System modeling:** How to abstract and further model the networked resources and related networking environment for distributed pervasive computing?
- **Interoperability:** How to actually overcome the heterogeneity of the pervasive computing environment, including middleware heterogeneity?
- **Networking:** How to benefit from the rich wireless networking technologies?
- **Quality of service:** How to effectively master the high dynamics and openness of pervasive computing environments and, in particular, how to ensure dependability and performance in such environments?

All the above mentioned challenges are inter-related, calling for their study in both the software engineering and distributed systems domains. Indeed, proposed middleware abstractions and related programming models shall effectively foster the development of *robust* distributed software systems, which, at the same time, must be implemented in an *efficient* way. Specifically:

- In the **software engineering domain**, ARLES studies resource abstractions and interaction paradigms to be offered by middleware, together with the associated languages, methods and tools for describing and composing the abstracted resources into applications. Our primary goal is to foster the development of *robust* distributed systems that are highly dynamic to adapt to the ever changing networking environment, and further meet quality of service requirements.
- In the **distributed systems domain**, ARLES studies innovative middleware architectures and related distributed algorithms and protocols for the *efficient* networking of distributed resources into distributed pervasive systems. Our further goal is to assist the abovementioned engineering process.

## 2.2. Highlights of the Year

During this year, we have provided complete solutions, including the development of models, algorithms and implementation/middleware for:

- Service-oriented middleware for pervasive networking so as to allow effectively exploiting the rich multi-radio, multi-network environments that are now accessible from most wireless devices (§ 6.2),
- Semantic service-oriented middleware so as to enable truly pervasive services, which may be discovered, accessed and composed anytime, anywhere (§ 6.3),

[1] V. Issarny *et al*. Systematic Aid for Developing Middleware Architectures. In Communications of the ACM, Special Issue on Adaptive Middleware, 45(6). 2002.

[2] M. Weiser. The Computer for the Twenty-First Century. In Scientific American. 1991.

[3] M. Satyanarayanan. Pervasive Computing: Vision and Challenges. In IEEE Personal Communications. August 2001.

[4] E. Aarts *et al.* Ambient Intelligence. In The Invisible Future: the Seamless Integration of Technology into Everyday Life. McGraw-Hill. 2002.

- Privacy awareness in the design of distributed systems for ambient intelligence, considering more specifically the enhancement of service-oriented middleware functions from the standpoint of privacy (§ 6.5), and

# 3. Scientific Foundations

## 3.1. Introduction

**Keywords:** *Web services*, *ambient intelligence*, *dependability*, *distributed systems*, *middleware*, *pervasive computing*, *service-oriented architecture*, *software architecture*, *software engineering*, *system composition*, *wireless networks*.

Research undertaken within the ARLES project-team aims to offer comprehensive solutions to support the development of pervasive computing systems that are dynamically composed according to the environment. This leads us to investigate dedicated software architecture styles with a special emphasis on associated:

- *architectural models* for mobile distributed software systems, enabling pervasive computing, together with associated methods and tools for reasoning about the systems' behavior and automating the systems' composition at runtime, and
- *middleware platforms* for alleviating the complexity of systems development, by in particular offering adequate network abstractions.

The next section provides a brief overview of the state of the art in the area of software architectures for distributed systems; we survey base architectural styles that we consider in our work and further discuss the benefits of architecture-based development of distributed systems. Section 3.3 then addresses middleware architectures for mobile systems, discussing the impact of today's wireless networks on the software systems, and core requirements that we consider for the middleware, i.e., managing the network's dynamics and enforcing dependability for the mobile systems. Each section refers to results on which we build, and additionally discusses some of the research challenges that remain in the area and that we are investigating as part of our research.

## 3.2. Software Architectures for Distributed Systems

Architectural representations of systems have shown to be effective in assisting the understanding of broader system concerns by abstracting away from details of the system. This is achieved by employing architectural styles that are appropriate for describing systems in terms of *components*, the interactions between these components – *connectors* – and the properties that regulate the composition of components – *configurations*. Thus, components are units of computation or data store, while connectors are units of interaction among components or rules that govern the interactions. Defining notations for the description of software architectures has been one of the most active areas of research in the software architecture community since its emergence in the early 90's. Regarding the overall development process, *Architecture Description Languages* (ADLs) that have been proposed so far are mainly concerned with architecture modeling during the analysis and design phase. In addition, some existing ADLs enable the deriving of system implementation and deployment, provided there is an available implementation of the system's primitive components and connectors. In general, a major objective in the definition of ADLs is to provide associated Computer-aided Software Engineering (CASE) tools, which enables tasks underpinning the development process to be automated. In this context, special emphasis has been put on the usage of formal methods and associated tools for the analysis of complex software systems by focusing on the system's architecture, which is abstract and concise. As a result, work in the software architecture community provides a sound basis towards assisting the development of robust distributed systems, which is further eased by middleware platforms.

### 3.2.1. Middleware-based and service-oriented software architectures

Available middleware can be classified into three main categories: transaction-oriented middleware that mainly aims at system architectures whose components are database applications; message-oriented middleware that targets system architectures whose component interactions rely on publish/subscribe communication schemes; and object-oriented middleware that is based on the remote procedure call paradigm and enables the development of system architectures complying with the object paradigm (*e.g.*, inheritance, state encapsulation), and, hence, enforces an object model for the system (i.e., the architectural components are objects). Development of middleware-based systems is now quite mature although middleware heterogeneity is still an open issue. In addition, dealing with middleware heterogeneity in the presence of dynamic composition raises the issue of dynamically integrating and possibly adapting the system's components, which is being investigated in the middleware community.

Evolution of middleware and distributed system technologies has further led to the emergence of service-oriented system architectures to cope with the requirements of Internet-based systems. *Software services*, in particular in the form of XML *Web services*, offer a promising paradigm for software integration and interoperation. Simply stated, a service is an instantiated configured system, which may be composed with other services to offer a new system that actually realizes a system of systems. Although the definition of the overall Web services architecture is still incomplete, the base standards defining a core middleware for Web services have already been released by the W3C[5], partly building upon results from object-based and component-based middleware technologies. These standards relate to the specification of Web services and supporting interaction protocols. Composing Web services relates to dealing with the assemblage of existing services, so as to deliver a new service, given the corresponding published interfaces. Integration of Web services is then realized according to the specification of the overall process composing the Web services. The process specifying the composition must not solely define the functional behavior of the process in terms of interactions with the composed services, but also the process' non-functional properties, possibly exploiting middleware-related services. Various non-functional properties (*e.g.*, availability, extensibility, reliability, openness, performance, security, scalability) should be accounted for in the context of Web services. However, enforcing dependability of composite Web services is one of the most challenging issues, especially for supporting business processes, due to the fact that the composition process deals with the assemblage of loosely-coupled autonomous components.

Although Web services have primarily been designed for realizing complex business processes over the Internet, they are a promising architectural choice for pervasive computing. The pervasiveness of the Web allows anticipating the availability of Web services in most environments, considering further that they may be hosted on mobile devices. Hence, this serves as a sound basis towards dealing with the dynamic composition of services in the pervasive computing environment. However, this further requires specification of the Web services' functional and non-functional behavior that can be exploited for their dynamic selection and integration, which may in particular build upon work on the Semantic Web.

### 3.2.2. Architecture-based development of distributed systems

The building blocks of distributed software systems relying on some middleware platform, fit quite naturally with the ones of software architectures: the architectural components correspond to the application components managed by the middleware, and the architectural connectors correspond to the supporting middleware. Hence, the development of such systems can be assisted with an architecture-based development process in a straightforward way. This approach is already supported by a number of ADL-based development environments targeting system construction, such as the Aster environment that was previously developed by members of the ARLES project-team.

However, most of the work on the specification of connectors has focused on the characterization of the interaction protocols among components, whilst connectors abstracting middleware embed additional complex functionalities (*e.g.*, support for provisioning fault tolerance, security, transactions). The above concern has led the software architecture community to examine the specification of the non-functional properties offered by

[5] http://www.w3.org

connectors. For instance, these may be specified in terms of logic formulae, which further enables synthesizing middleware customized to the application requirements, as supported by the Aster ADL. Another issue that arises when integrating existing components, as provided by middleware platforms, results from assembling components that rely on distinct interaction patterns. This aspect is known as *architectural mismatch* and is one of the criteria substantiating the need for connectors as first-class entities in architecture descriptions. The abstract specification of connector behavior, as, for instance, supported by the Wright ADL, enables reasoning about the correctness of component and connector composition with respect to the interaction protocols that are used. However, from a more pragmatic standpoint, software development is greatly eased when supported by mechanisms for solving architectural mismatches, which further promotes software reuse.

Connectors that are implemented using middleware platforms actually abstract complex software systems comprising a broker, proxies, but also services for enhanced distribution management. Hence, middleware design deserves as much attention as the overall system design, and must not be treated as a minor task. Architecture-based design is again of significant assistance here. In particular, existing ADLs enable describing conveniently middleware architectures. In addition, given the fact that middleware architectures build upon well known solutions regarding the enforcement of non-functional properties, the synthesis of middleware architectures that comply with the requirements of given applications may be partly automated through a repository of known middleware architectures. In the same way, this *a priori* knowledge about middleware architectures enables one to deal with the safe dynamic evolution of the middleware architectures according to environmental changes, by exploiting both the *support for adaptation* offered by middleware platforms (*e.g.*, reflective middleware) and the *rigorous specification of software architectures* enabled by ADLs.

As briefly outlined above, results on software architectures for distributed systems primarily lie in the definition of ADLs that allow the rigorous specification of the elements composing a system architecture, which may be exploited for the system's design and, further, for the software system's assessment and construction. Work in this area also includes closer coupling with solutions that are used in practice for the development of software systems such as integration of ADLs with the now widely accepted UML standard for system modeling. Still in this direction, coupling with OMG's Model-Driven Architecture (MDA) is much beneficial. Another area that has already deserved a great deal of attention in architecture-based development is the one of easing the design and construction of middleware underpinning the system execution out of existing middleware platforms. However, addressing all the features enabled by middleware within the architecture design is not yet fully covered. For instance, this requires reasoning about the composition of, possibly interfering, middleware services enforcing distinct non-functional properties. Another area of ongoing research work from the standpoint of architecture specification relates to handling needed architectural evolution as required by emerging applications, including those based on the Internet and/or aimed at mobile computing. In this context, it is mandatory to support the development of system architectures that can adapt to the environment. As a result, the system architecture shall handle dealing with the system evolution at runtime and further assessing the behavior of the resulting system.

## 3.3. Middleware Architectures for Mobile Systems

Advances in wireless networking combined with increasingly small-scale wireless devices are at the heart of the ambient intelligence (and pervasive computing) vision as they together enable ubiquitous networking and computing. However, developing software systems such that they can actually be accessed anywhere, anytime, while supporting natural interaction with users, remains a challenge. Although solutions to mobile/nomadic computing have now been investigated for more than a decade following the emergence of wireless networks and devices, these have mostly concentrated on adapting existing distributed systems architectures, so that the systems can tolerate the occurrence of disconnection. Basically, this had led to applying replication strategies to the mobile environment, where computation and/or data are cached on mobile nodes and later synchronized with peer replicas when connection allows.

Today's wireless networks enable dynamically setting up temporary networks among mobile nodes for the realization of some distributed function. However, this requires adequate development support, and in particular supporting middleware platforms for alleviating the complexity associated with the management of

dynamic networks. In this context, ad hoc networking is amongst the most challenging network paradigm for distributed systems, due to its highly dynamic topology and the absence of any infrastructure. Moreover, it offers significant advantages towards the realization of ubiquitous networking and computing, still due to the absence of any infrastructure. The following section provides a brief overview of ad hoc networking, and is then followed by an overview of the key middleware functionalities that we are addressing for assisting the development of mobile systems. Such functionalities relate to the management of the network's dynamics and to enforcing system dependability.

### *3.3.1. Ad hoc networking*

There exist two different ways of configuring a mobile network: infrastructure-based and ad-hoc-based. The former type of network structure is the most prominent, as it is in particular used in both Wireless LANs (*e.g.*, IEEE 802.11) and global wireless networks (*e.g.*, GSM, GPRS, UMTS). An infrastructure-based wireless network uses fixed network access points (known as base stations), with which mobile terminals interact for communicating, i.e., a base station forwards messages that are sent/received by mobile terminals. One limitation of the infrastructure-based configuration is that base stations constitute bottlenecks. In addition, it requires that any mobile terminal be in the communication range of a base station. The ad-hoc-based network structure alleviates this problem by enabling mobile terminals to cooperatively form a dynamic and temporary network without any pre-existing infrastructure.

The main issue to be addressed in the design of an ad hoc (network) routing protocol is to compute an optimal communication path between any two mobile terminals. This computation must minimize the number of control messages that are exchanged among mobile terminals, in order to avoid network congestion, but also to minimize energy consumption. There exist two base types of ad hoc routing protocols: proactive and reactive. Proactive protocols update their routing table periodically. Compared to proactive protocols, reactive protocols *a priori* reduce the network load produced by the traffic of control messages, by checking the validity of, and possibly computing, the communication path between any two mobile terminals only when communication is requested between the two. Hybrid routing protocols further combine the reactive and proactive modes. The design rationale of hybrid protocols is that it is considered advantageous to accurately know only the neighbors of any mobile terminal. Since they are close to the terminal, communicating with neighbors is less expensive, and neighbors are most likely to take part in the routing of the messages sent from the terminal. Based on this, a hybrid protocol implements: (i) a proactive protocol for communication with mobile terminals in the neighborhood, and (ii) a reactive protocol for communication with the other terminals.

Spurred by the progress of technologies and deployment at low cost, the use of ad hoc networks is expected to be largely exploited for mobile computing, and no longer be restricted to specific applications (i.e., crisis applications as in military and emergency/rescue operations or disaster recovery). In particular, ad hoc networks effectively support ubiquitous networking, providing users with network access in most situations. However, we do not consider that pure ad hoc networks will be the prominent wireless networks. Instead, mobile distributed systems shall be deployed on hybrid networks, combining infrastructure-based and ad hoc networks, so as to benefit from their respective advantages. Development of distributed systems over hybrid wireless networks remains an open challenge, which requires dedicated middleware solutions for in particular managing the network's dynamics and resources.

### *3.3.2. Managing the network's dynamics*

Trends in mobile computing have created new requirements for automatic configuration and reconfiguration of networked devices and services. This has led to a variety of protocols for lookup and discovery of networked resources. In particular, *discovery protocols* provide proactive mechanisms for dynamically discovering, selecting and accessing available resources. As such, resource discovery protocols constitute a core middleware functionality towards managing the network's dynamics in mobile computing systems. Resource discovery is a central component of distributed systems as it enables services and resources to discover each other on a network and evaluate potential interactions. Many academic and industry-supported protocols (*e.g.*, SLP, UDDI, SSDP) have been designed in different settings, and numerous are now in common usage, using either distributed or centralized approaches depending on assumptions about the underlying network and the

environment. These design constraints have led to different, sometimes incompatible mechanisms for service advertisements, queries, security and/or access, while none of the existing resource discovery protocols is suitable for all environments.

The major structural difference between existing resource discovery protocols is the reliance (or not) on a central directory. A central directory stores all the information concerning resources available in the network, provided that resources advertise themselves to the central directory using a unicast message. Then, to access a resource, a client first contacts the central directory to obtain the resource's description, which is to be used for contacting the resource's provider. Prior to any resource registration or client request to the central directory, clients and resource providers must first discover the central directory by issuing broadcast or multicast requests. Centralized resource discovery is much suited to wireless infrastructure-based networks. However, this makes the discovery process dependent upon the availability of the central directory, which further constitutes a bottleneck. In order to support resource discovery in a wider network area, the use of a distributed set of fixed directories has been proposed. Directories are deployed on base stations (or gateways) and each one is responsible for a given discovery domain (*e.g.*, corresponding to a cell).

In the self-organizing wireless network model provided by ad hoc networks that use peer-to-peer communication and no fixed infrastructure, the use of fixed directories for resource discovery is no longer suitable. In particular, the selection of mobile terminals for hosting directories within an ad hoc network is a difficult task, since the network's topology frequently changes, and hence the connectivity is highly dynamic. Decentralized resource discovery protocols then appear more suitable for ad hoc networks. In this case, resource providers and clients discover each other directly, without interacting with a central directory. Specifically, when a client wants to access a resource, it sends a request to available providers using a broadcast message. However, this approach leads to the flooding of the network. An approach to disseminating information about network resources while not relying on the use of broadcast is to use geographic information for routing. Nodes periodically send advertisement along a geometric trajectory (basically north-south and west-east), and nodes located on the trajectory both cache and forward advertisements. Then, when a client seeks a resource, it sends a query that eventually intersects an advertisement path at a node that replies to the request. This solution assumes that the density of nodes is high enough, and further requires the replication of resource advertisements on a significant number of nodes. Hence, it incurs resource consumption that may not be accommodated by wireless, resource-constrained nodes. Resource consumption is further increased by the required support for geographical location (*e.g.*, GPS). Other solutions to decentralized resource discovery that try to minimize network flooding are based on local resource discovery. Broadcast is limited to the neighborhood, hence allowing only for resource discovery in the local area, as supported by base centralized resource discovery protocols. Discovery in the wider area then exploits solutions based on a hierarchy of discovery domains.

Resource discovery protocols for hybrid networks that in particular suit ad hoc networks remains an open issue. Other fundamental limitations of the leading resource discovery protocols are: (i) reliance on syntactic matching of resource attributes included in the resource description, and (ii) unawareness of the environment where the resources are provided. The development of mobile/handheld devices, and wireless and ad hoc networks (*e.g.*, Wi-Fi, Bluetooth) have enabled the emergence of service-rich environments aimed at supporting users in their daily life. In these pervasive environments, a variety of infrastructure-based and/or infrastructure-less networks are available to the users at a location. Such heterogeneous environments bring new challenges to resource/service discovery, including context-awareness, protocol interoperability, semantic knowledge, etc.

While resource discovery constitutes a core middleware functionality towards easing the development of distributed software systems on top of dynamic networks, higher-level abstractions for dynamic networks need to be developed and supported by the middleware for easing the developers' task. The definition of such abstractions shall be derived from both features of the network and architectural principles elicited for mobile software systems, where we exploit our work in both areas. Related issues include characterizing and reasoning about the functional and non-functional behavior of the participating peer nodes, and in particular dealing with security requirements and resource availability that are crucial in the mobile environment.

### *3.3.3. Enforcing dependability*

Dependability of a system is defined as the reliance that can justifiably be placed on the service that the system delivers. It decomposes into properties of availability, safety, reliability, confidentiality, integrity and maintainability, with security encompassing availability, confidentiality and integrity. Dependability affects the overall development process, combining four basic means that are fault prevention, fault removal, fault tolerance and fault forecasting. In the context of middleware architectures for mobile systems, we concentrate more specifically on fault tolerance means towards handling mobility-induced failures. Such failures affect most dependability properties. However, availability and security-related properties are the most impacted by the mobile environment due to changing connectivity and features of wireless networks that make them more prone to attacks. Security remains one of the key challenges for mobile distributed systems. In particular, the exploitation of ad hoc networks does not allow systematic reliance on a central infrastructure for securing the network, calling for decentralized trust management. Additionally, resource constraints of mobile devices necessitate the design of adequate cryptographic protocols to minimize associated computation and communication costs.

Enforcing availability in the mobile environment relies on adequate replication management so that data and/or services remain accessible despite the occurrence of disconnection. Such a concern has led to tremendous research work since the emergence of mobile computing. In particular, data replication over mobile nodes has led to novel coherency mechanisms adapted to the specifics of wireless networks. Solutions in the area relate to offering optimistic coherency protocols, so that data copies may be concurrently updated and later synchronized, when connectivity allows. In initial proposals, data copies were created locally on accessing nodes, since these proposals were aimed at global infrastructure-based networks, where the mobile node either has access to the data server or is isolated. However, today's wireless networks and in particular ad hoc networks allow for creating temporary collaborative networks, where peer nodes may share resources, provided they trust each other. Hence, this allows addressing the replication of data and services over mobile nodes in accordance with their respective capabilities. Dually, peer-to-peer communication supported by ad hoc networks combined with decentralized resource discovery allow accessing various instances of a given resource, and hence may be conveniently exploited towards increasing availability. Today's wireless networks offer great opportunities towards availability management in mobile systems. However, providing effective solutions remains an open issue, as this must be addressed in a way that accounts for the constraints of the environment, including possible resource constraints of mobile nodes and changing network topology. Additionally, solutions based on resource sharing among mobile nodes require incentive mechanisms to avoid selfish behavior where nodes are trying to gain but not provide resource access.

# 4. Application Domains

## 4.1. Application Domains

**Keywords:** *Web services*, *ambient intelligence*, *distributed systems*, *information systems*, *mobile systems*, *pervasive computing*.

The ARLES project-team targets development support for applications relevant to the ambient intelligence/pervasive computing vision, with a special focus on consumer-oriented applications. Architecture-based development of composite systems is further directly relevant to enterprise information systems, whose composition is mainly static and relates to the integration of legacy systems. In addition, by building upon the Web services architecture for dealing with the dynamic composition of (possibly mobile) autonomous systems, our work is of direct relevance to e-business applications, providing specific solutions for the mobile context.

Our application domain is voluntarily broad since we aim at offering generic solutions. However, we examine exploitation of our results for specific applications, as part of the experiments that we undertake to validate our research results through prototype implementation. Applications that we consider in particular include demonstrators developed in the context of the European and National projects to which we contribute (§ 7.1 & 7.3).

As an illustration of applications investigated within ARLES, the COCOA semantic service middleware together with the INMIDIO interoperable middleware (respectively, § 5.6 & 5.5) support the *networked home* environment. The networked home specifically seeks to combine the home automation, consumer electronics, mobile communications and personal computing domains to provide new user applications that exploit the fluid integration of these traditionally strictly separated domains, and to lay a solid foundation towards realizing the ambient intelligence vision.

# 5. Software

## 5.1. Introduction

In order to validate our research results, our research activities encompass the development of related prototypes. We present here chronologically the prototypes that are released under open source license at http://www-rocq.inria.fr/arles/doc/doc.html.

## 5.2. WSAMI: A Middleware Based on Web Services for Ambient Intelligence

**Participant:** Valérie Issarny [correspondent].

WSAMI (Web Services for AMbient Intelligence) is based on the Web services architecture and allows for the deployment of services on wireless handheld devices like smartphones and PDAs [7]. WSAMI further supports the dynamic composition of distributed services over hybrid wireless networks. Moreover, WSAMI takes in charge the customization of the network's path through the dynamic integration of middleware-related services, in order to enforce quality of service with respect to offered dependability and performance properties.

The WSAMI middleware prototype is available since 2004. It is a Java-based implementation of the WSAMI core middleware, which builds upon IEEE 802.11b as the underlying WLAN and integrates the following components: (i) the WSAMI SOAP-based core broker, including the CSOAP[6] SOAP container for wireless, resource-constrained devices; and (ii) the Naming & Discovery service, including support for connector customization. The memory footprint of our CSOAP implementation is 90kB, as opposed to the 1100kB of the Sun's reference SOAP implementation. The overall memory footprint of our Web services platform is 3.9MB, dividing into 3MB for the CVM and 815kB for the Xerces XML parser, in addition to the CSOAP implementation. The WSAMI middleware prototype is an open-source software freely distributed since 2004 under the terms of the GNU Lesser Public License (LGPL) at http://www-rocq.inria.fr/arles/download/ozone/index.htm.

Our prototype is being used for the implementation of demonstrator applications in the field of ambient intelligence, as well as a core for service-oriented middleware platforms aimed at advanced wireless networking environments, like Ariadne, presented below.

## 5.3. Ariadne: A Protocol for Scalable Service Discovery in MANETs

**Participant:** Valérie Issarny [correspondent].

The Ariadne service discovery protocol has been designed to support decentralized Web service discovery in multi-hop mobile ad hoc networks (MANETs) [10]. Ariadne enables small and resource-constrained mobile devices to seek and find complementary, possibly mobile, Web services needed to complete specified tasks, while minimizing the traffic generated and tolerating intermittent connectivity. The Ariadne protocol further enables service requesters to differentiate services instances according to non-functional properties. Specifically, the Ariadne protocol is based on the homogeneous and dynamic deployment of cooperating directories within the MANET. Scalability is achieved by limiting the generated traffic related to service discovery, and by using compact directory summaries (i.e., Bloom filters) to efficiently locate the directory that most likely has the description of a given service.

[6] Compact SOAP

The prototype of the Ariadne service discovery protocol has been implemented in Java, and provides an application programming interface (API) so as to be easily integrated in a Web service-oriented middleware such as WSAMI, presented above. The Ariadne prototype is an open source software freely distributed since 2005 under the terms of the GNU Lesser Public License (LGPL) at http://www-rocq.inria.fr/arles/download/ariadne/index.html.

Our prototype is being used for the implementation of demonstrator applications exploiting MANETs in the field of ambient intelligence.

## 5.4. MUSDAC: A Middleware for Service Discovery and Access in Pervasive Networks

**Participant:** Valérie Issarny [correspondent].

The MUlti-protocol Service Discovery and ACcess (MUSDAC) middleware platform enables the discovery and access to services in the pervasive environment, which is viewed as a loose and dynamic composition of independent networks [8]. MUSDAC manages the efficient dissemination of discovery requests between the different networks and relies on specific plug-ins to interact with the various middleware used by the networked services.

We have implemented a first prototype of MUSDAC in Java (J2SE 1.4.2 and 1.5), which includes support for 5 different service discovery protocols, and remote access for SOAP-based services. The different plug-ins enable us to experiment with both repository-based (Ariadne, OSGi) and multicast-based (SLP, UPnP) protocols. The MUSDAC prototype is an open source software freely distributed since 2005 under the terms of the GNU Lesser Public License (LGPL) at http://www-rocq.inria.fr/arles/download/ubisec/index.html.

The MUSDAC middleware in particular serves as a base building block in the development of a middleware aimed at effectively enabling service-oriented computing in Beyond 3rd Generation (B3G) networks. This further led to make evolve MUSDAC-based service discovery and access to multi-radio and multi-protocol networking environments (see § 5.7).

## 5.5. INMIDIO: An Interoperable Middleware for Ambient Intelligence

**Participant:** Nikolaos Georgantas [correspondent].

In the pervasive computing environment, devices from various application domains, *e.g.*, home automation, consumer electronics, mobile and personal computing domains, need to dynamically interoperate irrespective of the heterogeneity of their underlying hardware and software. Middleware has been introduced in order to overcome this issue by specifying a reference interaction protocol enabling compliant software systems to interoperate. However the emergence of different middleware platforms to address the requirements of specific application domains leads to a new heterogeneity issue among interaction protocols. Thus, at a given time and/or at a specific place, devices hosting the wrong middleware become isolated. In order to overcome this issue, we have developed a system called INMIDIO (INteroperable MIddleware for service Discovery and service InteractiOn) that dynamically resolves middleware mismatch. More particularly, INMIDIO identifies the interaction middleware and also the discovery protocols that execute on the network and translates the incoming/outgoing messages of one protocol into messages of another, target protocol [3]. Specifically, the system parses the incoming/outgoing message and, after having interpreted the semantics of the message, it generates a list of semantic events and uses this list to reconstruct a message for the target protocol, matching the semantics of the original message. The INMIDIO middleware acts in a transparent way with regard to discovery and interaction middleware protocols and with regard to the services running on top of them.

The service discovery protocols supported by the current INMIDIO prototype are UPnP, SLP and WS-Discovery, while the supported service interaction protocols are SOAP and RMI. The INMIDIO prototype is publicly available since 2006 and released under the GNU Lesser Public License (LGPL) at http://www-rocq.inria.fr/arles/download/inmidio/index.html.

## 5.6. COCOA: A Semantic Service Middleware

**Participant:** Nikolaos Georgantas [correspondent].

COCOA is a comprehensive approach to semantic service description, discovery, composition, adaptation and execution, which enables the integration of heterogeneous services of the pervasive environment into complex user tasks based on their abstract specification [1], [2]. Using COCOA, abstract user tasks are realized by dynamically composing the capabilities of services that are currently available in the environment. The capabilities that a service provides are presented as a *conversation* – a workflow that specifies data and control dependencies for its capabilities. Similarly, an abstract task is presented as an orchestration of required capabilities. The conversations of the provided service capabilities are integrated to realize the orchestration of the abstract task, while guaranteeing that the dependencies of each of the provided capabilities are preserved. This allows complex user tasks to be created and reliably composed, while offering fine-grained control over the placement of capabilities in the task. This is especially important for the pervasive environment, where user tasks may frequently involve interaction with the user(s). In addition, the service composition can be optimized based on quality of service and context-aware parameters. To accommodate the inherent heterogeneity of services in the pervasive environment, capabilities are described and matched semantically, and adapted when necessary. Furthermore, different service groundings are supported, allowing diverse SOA platforms to be incorporated; interoperability at service grounding, *i.e.*, middleware level may then be ensured by employing INMIDIO (§ 5.5). Once a new service realizing the user task has been created, it is automatically deployed and executed.

A prototype version of COCOA has been released under LGPL as open source software in 2007 on the Amigo GForge Open Source Software site http://gforge.inria.fr/projects/amigo/.

## 5.7. ubiSOAP: A Service Oriented Middleware for Seamless Networking

**Participant:** Valérie Issarny [correspondent].

The ubiSOAP middleware empowers the service-oriented architecture with pervasive networking capabilities, in particular enabling adaptive lightweight services to be run on mobile nodes and access to services over multi-radio, multi-network links [4].

The ubiSOAP middleware specifically enriches the Web service architecture with key features for services to become truly pervasive by taking full benefit of the rich capacities, including multi-radio interfaces, now embedded in wireless devices. Specifically, ubiSOAP brings multi-radio, multi-network connectivity to services through a comprehensive layered architecture:

- The multi-radio device management and networking layers together abstract multi-radio connectivity, selecting the optimal communication link to/from nodes, according to quality parameters.
- The communication layer allows for communication in the pervasive networking environment according to SOAP. ubiSOAP in particular enriches traditional functionalities of a SOAP engine to allow for SOAP-based point-to-point and group-based interactions in the pervasive network. It further enables access to services that may be in distinct networks thanks to multi-network routing.
- The middleware services layer brings advanced distributed resource management functionalities customized for the pervasive networking environment. From that layer, the Discovery Service enables the dynamic advertising and location of networked services, in particular accounting for extra-functional properties.

Last but not least, the ubiSOAP middleware is mobility-aware so that its functionalities adapt to the physical mobility of both clients and services, in particular exploiting the rich multi-radio, multi-network connectivity.

The ubiSOAP middleware has been implemented using Java for both desktop (J2SE) and mobile (J2ME CDC) environments. The software has been developed in the context of the IST PLASTIC Project and can be freely downloaded since 2008 under the GNU LGPL license at http://www.ist-plastic.org.

# 6. New Results

## 6.1. Introduction

The ARLES project-team investigates solutions in the forms of languages, methods, tools and supporting middleware, to assist the development of eistributed software systems, with a special emphasis on mobile distributed systems enabling the ambient intelligence/pervasive computing vision. We in particular study ambient intelligence system architectures based on the service paradigm, from service modeling to service-oriented middleware for advanced wireless networks. Our research activities over the year 2008 have focused on the following complementary areas:

- Service-oriented middleware for pervasive networking so as to allow effectively exploiting the rich multi-radio, multi-network environments that are now accessible from most wireless devices (§ 6.2),
- Semantic service-oriented middleware so as to enable truly pervasive services, which may be discovered, accessed and composed anytime, anywhere (§ 6.3),
- Automatic service composition so as to enable distributed software systems to be dynamically composed out of networked resources (§ 6.4),
- Privacy awareness in the design of distributed systems for ambient intelligence, considering more specifically the enhancement of service-oriented middleware functions from the standpoint of privacy (§ 6.5), and
- Middleware support to allow ubiquitous access of user data from a multitude of devices with heterogeneous capabilities and running on different platforms (§ 6.6).

## 6.2. Service-oriented Middleware for Pervasive Networking

**Participants:** Mauro Caporuscio, Valérie Issarny, Hassine Moungla, Animesh Pathak, Pierre-Guillaume Raverdy.

With network connectivity being embedded in most computing devices, networking environments are now pervasive. As a result, any networked device may not only seamlessly consume but also provide software applications over the network. Service-Oriented Computing (SOC) then introduces natural design abstractions to deal with pervasive networking environments. Indeed, networked software applications may conveniently be abstracted as autonomous loosely coupled services, which may be combined to accomplish complex tasks. In addition, the concrete instantiation of SOC paradigms provided by Web Services (WS) technologies by means of Web-based/XML-based open standards (e.g. WSDL, UDDI, HTTP, SOAP) may be exploited for concrete implementation of pervasive services. However, while Web services standards and implementations targeting wide-area domains are effective technologies, supporting Web service access in pervasive networking environments is still challenging. In such kind of networking environments, mobile applications, acting as both service consumers and providers, often run on scarce resource platforms such as personal digital assistants and mobile phones, which have limited CPU power, memory, and battery life. Moreover, these devices are usually interconnected through one or more heterogeneous wireless links, which compared to wired networks are characterized by lower bandwidths, higher error rates, and frequent disconnections.

A key feature of pervasive networking environments is the diversity of radio links available on portable devices, which may be exploited towards seamless connectivity. Specifically, as nodes get connected via multiple radio links, thorough scheduling and handover across those links allows enhancing overall connectivity and actually making it seamless. This calls for making services network-agnostic, so that the underlying middleware takes care of scheduling exchanged messages over the embedded links in a way that best matches Quality of Service (QoS) requirements, and further ensures service continuity through vertical handover. In this setting, a primary requirement for supporting service-oriented middleware is to provide a comprehensive networking abstraction that allows applications to be unaware of the actual underlying networks while still exploiting their diversities in terms of both functional and extra-functional properties.

To address the above, we have introduced the ubiSOAP communication middleware [18], which underlies SOAP-based middleware and strives to provide pervasive networking to services. As sketched below, the architecture of ubiSOAP layers the following constituents below SOAP-based middleware functionalities: (i) multi-radio networking provides network-agnostic connectivity, (ii) multi-network routing implements a multi-network overlay, and (iii) SOAP Communication leverages multi-radio, multi-network message routing, and further introduces communication primitives targeted at pervasive computing systems. As discussed in Section 5.7, the prototype implementation of ubiSOAP is released under open source license.

**Multi-Radio Networks.** The *multi-radio networking layer* of ubiSOAP provides core functionalities to effectively manage multi-radio connectivity by providing a network-agnostic addressing scheme together with QoS-aware network link selection. In particular, the *multi-radio networking layer* is in charge of:

- *Network-agnostic addressing:* Devices embedding multiple network interfaces may have multiple IP addresses, at least one for each active interface. Thus, in order to identify uniquely a given application in the network we associate to it a *Multi-Radio Network Address* (MRN@). The MRN@ of an application instance is specifically the application's Unique ID, which resolves into the actual set of IP addresses (precisely, $network\_ID \oplus IP$ addresses) bound to the device (at a given time) that runs the given instance. Then, upper layers shall use MRN@ as part of their addressing scheme (e.g., through WS-addressing in the case of Web services), which replaces the traditional IP-based addressing scheme. MRN@s are automatically generated and managed by the multi-radio networking layer. Furthermore, the multi-radio networking layer allows for performing a lookup operation that, starting from an MRN@, returns the set of IP addresses actually bound to it.
- *QoS-aware interface activation:* Next to MRN@ addressing, it is crucial to activate and select the best possible networks (among those available) with respect to required QoS. Interface activation allows the user application to activate the best possible interfaces (among those available) with respect to the required QoS. In particular, the application submits its QoS requirement (specified as set of pairs $< QoS_{attribute}, QoS_{value} >$) to the multi-radio networking layer, which in turn compares it with the QoS of each available interface. In this case, since the interface is switched off, QoS refers to the theoretic values of a network interface declared by the manufacturer (e.g., GPRS maximum bitrate = 171.2Kb/s). If the interface satisfies the requirement posed by the application, within a given approximation expressed in percentage, it is activated. It is also possible to define priorities upon the various quantitative parameters, in order to specify if a given parameter is more important than the others.
- *QoS-aware network selection:* Network selection is performed during the establishment of the communication and takes into account the QoS attributes required by the client application that is initiating the connection, as well as the networks active on the server listening for incoming connections, as given by the server's MRN@. If the client and the server share only one network that satisfies the requirements, it is used to carry on the interaction. On the other hand, when the two parties share more than one network, the selection algorithm selects the one that best meets the required QoS.
- *Communication:* Base unicast and multicast communication schemes are provided on top of MRN@ network-agnostic addressing: (i) *Synchronous unicast* is provided by means of a packet input/output stream that is used to read/write packets to be exchanged during the interaction between client and server applications; and (ii) *Asynchronous multicast* allows the user application to send multicast packets to all members of a given group.

**Multi-network Overlay.** Thanks to the ubiSOAP multi-radio networking layer, communication among nodes exploits the various network links that the nodes have in common, further selecting the link that provides the required QoS. However, in some cases, it might also be desirable for nodes to be able to access services that are hosted in networks to which the requesting node is not directly connected (e.g., to provide continuity of service despite node mobility). For this purpose, ubiSOAP introduces an overlay network that bridges heterogeneous networks, thus enhancing overall service connectivity. Specifically, nodes that are connected to two (or more)

different networks through their network interfaces can assume the role of bridge nodes. Bridge nodes quite literally "bridge" between two separate networks, relaying point-to-point and multicast ubiSOAP messages across those networks. In particular the *multi-network routing* layer is in charge of:

- *Multi-network point-to-point routing:* In ubiSOAP multi-network, multi-radio environments, the $network\_ID \oplus IP$ address embedded in the MRN@ of a host contains, along with the network address of the service host, the network ID that uniquely identifies the network (e.g., a BSSID, MAC address of the Bluetooth master, etc.) that the host resides in under its given address. To achieve effective routing across bridges, we propose a straightforward approach based on the principle of Mobile Ad hoc NETwork (MANET) routing. In this approach, bridge nodes advertise their presence to the nodes in their corresponding networks and exchange routing information. For this purpose, bridge nodes run an instance of OLSR among each other. However, instead of concrete node addresses bridges store as destinations the identifiers of the various present networks (i.e., $network\_ID$) and as next hop the bridge that needs to be contacted next to eventually reach the target network.
- *Multi-network multicast routing:* It is crucial to support both point-to-point and group interactions in the multi-radio, multi-network environment. In particular, multicasting is central to advanced middleware services like dynamic discovery. We thus introduce multi-network multicast routing, building upon multicast facilities of the composed networks. Multicast routing is such that within an IP network, the network's multicast facility (i.e., IP multicast or higher level group communication like Java Groups) is used for communication among group members. Then, multicast messages are forwarded by ubiSOAP bridges up to a fixed number of hops, while avoiding cycles and duplication. The ubiSOAP multi-network routing facility enables the definition of application-level groups. Specifically, application-level groups are individually managed at the ubiSOAP middleware layer while a single ubiSOAP multi-network group is managed in the network layer. The latter is actually a composition of ubiSOAP multicast groups, one for each of the composed networks.

**Custom SOAP Transport for Pervasive Networking.** In order to leverage the provided multi-radio, multi-network service connectivity, ubiSOAP introduces custom SOAP *point-to-point communication* that brings multi-radio multi-network routing to legacy SOAP messaging. Further, ubiSOAP implements a SOAP transport for *group communication* that enhances the SOAP API to meet the corresponding base requirement of pervasive networking environments. In particular, the *SOAP Communication* layer is in charge of:

- *Point-to-point communication:* The ubiSOAP point-to-point transport is a connection-oriented transport for supporting communication between a client and a service. This transport interacts with the multi-radio networking layer to send and receive messages over the network based on the MRN@ that identifies the remote party. It also interacts with the SOAP engine or the client SOAP library to receive or dispatch SOAP messages locally. When sending a message, the ubiSOAP point-to-point transport must first evaluate if the destination is directly reachable (i.e., the MRN@ of the sender and of the destination share a common network). If true, the message is then sent directly to the destination. If not, the transport retrieves the MRN@ of a ubiSOAP bridge directly reachable, encapsulates as plain data the application's SOAP message into a specific forwarding message, and sends this forwarding message to the bridge. This message is forwarded between bridges until it reaches the destination where the application's SOAP message is extracted and dispatched.
- *Group communication:* The ubiSOAP group transport is a connectionless transport for one-way communication between multiple peers in multi-network configurations. The ubiSOAP group transport component interacts with the multi-radio networking layer to send group messages based on an MRN@ identifying the group, and with the SOAP engine to deliver the group's messages to the registered services. As noted above, groups are identified with an MRN@. Multicast-based applications usually assume that all group members agree beforehand on a specific IP address for the group. We therefore also assume that all group members use the same MRN@ for the group. While services are not able to directly return a result to a client (one-way multicast), a service may send a message (one-way unicast) on the group directed at a specific peer. As group communication in

the underlying multi-radio networking layer is multicast-based, it does not guarantee the ordering or the delivery of messages. While ordering may be easily achieved on the receiving side, the overhead to provide group reliability is deemed too costly due to the dynamics of pervasive networks. Also, while many mobile devices may run the same collaborative application, a user may only be interested in interacting with the ones at its location. Such scoping may be achieved by limiting the forwarding of group's messages or by adding forwarding constraints.

**Service Discovery over ubiSOAP.** To validate the above, we have realized a Pervasive Service Discovery (PSD) Service that provides dynamic, interoperable, context-aware service discovery by benefiting from all the advanced features of ubiSOAP, i.e., group communication, and multi-radio, multi-network message routing. PSD is mainly a reengineering of the open source MUSDAC multi-protocol service discovery platform (see § 5.4) on top of ubiSOAP in order to support service discovery in multi-radio, multi-network environments.

PSD uses a hierarchical approach for service discovery in multi-network environments. Indeed, a (logically) centralized repository (PSD-S) coordinates service discovery within an independent network, while PSD-Ss in different networks communicate together in a fully distributed way to disseminate service information. While in MUSDAC service discovery and access were tightly integrated, PSD-Ss are only concerned with service discovery, and rely on ubiSOAP group communication to disseminate service information across networks. Changes in the multi-network topology (e.g., broken propagation paths) are then taken care of transparently.

PSD-Ss provide an explicit API supported by *PSD plugins* that enables clients (resp. providers) in a network to discover (resp. advertise) a service in the multi-network environment. It further enables clients and providers to benefit from advanced discovery features (e.g., context-awareness) by directly issuing requests or advertisements in the PSDL format. Specific *legacy SDP plugins* register with the active SDPs in the network, and translate requests and advertisements in legacy formats to PSDL (e.g., SLP and UPnP). PSD combines various matching algorithms to support the various elements of the service description (for both requests and advertisements), and thus provides comprehensive interoperability between SDPs. Finally, PSD controls the dissemination of local requests and the compilation of the results returned by distant PSD-Ss.

As described above, the PSD repository stores PSDL service descriptions that are either generated by legacy SDP plugins (e.g., UPnP2PSD plugin) or directly registered by service providers using the PSD plugin. We use a hierarchical service description format that actually combines a number of distinct documents specifying different facets of the service. The PSDL description acts primarily as a top-level container for additional files describing facets of the service. For example, a WSDL document may be used to describe the service interface while non-functional properties can be described using existing QoS and context models. The ubiSOAP grounding of host, that identifies the networks and IP address at which the service's host is network-reachable (i.e., MRN@ and mapping) is described in such separate document thus facilitating dynamic updates.

## 6.3. Semantic Middleware for Service-Oriented Pervasive Computing

**Participants:** Sébastien Bianco, Oleg Davidyuk, Vasileios Fotopoulos, Manel Fredj, Nikolaos Georgantas, Valérie Issarny, Graham Thomson, Apostolos Zarras.

As discussed in the previous section, service-oriented middleware (SOM) appears to be most appropriate to tackle the requirements of pervasive environments by abstracting the software and hardware resources of such environments as services and by supporting service discovery, access and composition. Nevertheless, the affluence of SOM technologies and platforms that have been put forward to address the heterogeneity and dynamics of pervasive environments has engendered a new kind of heterogeneity, i.e., middleware heterogeneity. This heterogeneity concerns the protocols associated with base middleware functionalities, which are service discovery and service access, as well as the multitude of networks in which service providers and requesters may reside. Thus, a SOM for pervasive computing should provide multi-protocol and multi-network interoperability mechanisms (see the previous section). Still, even after interoperability has been established at the networking and middleware levels, the dynamic discovery and composition of networked services by applications further require service providers and requesters to agree on the semantics of services, so that these can integrate and interact in a way that guarantees dependable service

provisioning and consumption. Such an agreement cannot be carried out at the syntactic level, i.e., by assuming that service providers and requesters use a common syntax for denoting service semantics, as this is too strong an assumption for pervasive environments. Then, a promising approach towards addressing syntactic heterogeneity relies on semantic modeling of service features by employing relevant semantic technologies. Nevertheless, assessing the conformance between service semantics as announced by service providers and requested by service requesters induces costly semantic reasoning (in terms of time and computation), which makes existing solutions inappropriate for the highly interactive and resource constrained pervasive environment. Finally, besides dealing with the functional features of services, user-centrism of pervasive environments calls for the awareness of service non-functional features, i.e., Quality of Service (QoS). To address the above challenges, we have introduced an efficient, semantic, QoS-aware service-oriented middleware for pervasive computing which integrates advanced semantic service model for pervasive services, efficient and lightweight semantic service registry [2], and automatic service composition [1]. The resulting COCOA service-oriented middleware is further released under open source license (see § 5.6).

Looking further into the dynamics of pervasive services, we have studied the issue of dependability. Runtime service reconfiguration is put forward as one of the means by which we could provide dependable service-oriented architectures (SOA) that enable continuity in service provisioning, and are flexible and robust in the presence of change. Indeed, in pervasive computing environments networked entities join and leave the environment at their convenience without prior notification. This dynamics comes at the price of dependability, due to runtime variations in terms of (i) service availability, and (ii) network connection/infrastructure availability, according to user/service mobility.

In this context, the main focus of our work is to incorporate support for runtime reconfiguration in SOA systems in order to tolerate runtime variations and ensure continuity in service provisioning for the users. In particular, we focus on middleware support for runtime service reconfiguration, since compared to application-level or network-level approaches, middleware seems to provide the required level of abstraction and genericity to deal with the challenges posed. Our main contribution consists in enabling service continuity by (i) *substituting* a service that becomes unavailable at runtime with a semantically similar one, and (ii) *transferring* and *translating* the current state of interaction to the substitute service in order to resume the execution from the point it was interrupted. The need for state translation is due to the environments' heterogeneity, since the unavailable and substitute services are not assumed to be identically *implemented* and/or *described*.

The runtime reconfiguration thus includes automatically substituting an unavailable service and transferring its state in a way that is compatible with the substitute service. Ensuring compatibility includes:

1. Semantic matching between the unavailable service and the substitute one. To enable this, we establish matching relations building upon formal subtyping relations introduced in seminal work in the object-oriented domain.
2. Checking correctness of partial (i.e., at a certain point of execution) substitution between the two services. To enable this, we (a) formally define service state and state compatibility relations, and (b) establish behavior compatibility relations building upon seminal work on simulation between computer programs.

In the case where the unavailable sevice is part of an orchestrated service compition, the impact of substitution on the still available services of the composition should also be considered. Indeed, depending on the point where the orchestration execution was interrupted and the point of execution where state transfer is possible between the unavailable and the substitute service, a roll-back may be needed for the substitute service with respect to execution point where the unavailable service was interrupted. Then, due to data dependencies, some of the still available services of the composition may also have to roll back. To deal with this, we establish checkpointing and introduce algorithms for rollback computation.

The outcome of our contribution is SIROCCO (ServIce Reconfiguration upOn serviCe unavailability and Connectivity lOss), a middleware infrastructure that enables transparent runtime reconfiguration of SOA systems upon services unavailability. The middleware discovers candidate substitute services, which are semantically compatible services that can be used in place of the service that becomes unavailable, identifies

the best amongst these candidates and performs the substitution. In the case of service composition, the middleware also takes into account the impact on the still available services of the composition.

## 6.4. Automatic Service Composition

**Participants:** Sandrine Beauche, Nebil Ben Mabrouk, Nikolaos Georgantas, Pascal Poizat.

*Automatic service composition* is one of the most challenging issues in Service-Oriented Computing (SOC), as it enables to provide the end-users with functionalities composed out of services existing in their environment, or more generally to have services collaborating in added-value composite services. Automatic service composition techniques rely on four interface description levels: signature (operation profiles), behavior (protocols), non functional (time, QoS) and semantics. However, these techniques usually assume that services have been previously developed to be integrated, and the proposed composition processes are limited to simple correspondences between service functionalities.

*Software adaptation* has provided solutions for component interoperability through the computation – from component interfaces and user-defined adaptation abstract specifications, called mappings – of adaptors that operate in-between components to ensure their correct composition at the signature and behavioral levels. In [13] we have proposed a state-of-the art behavioral level adaptation technique based on transition systems and Petri net encodings to solve mismatch between components. Further, different studies have been undertaken to make adaptation applicable to Service-Oriented Architectures (SOA). This followed three complementary directions: adaptation of service orchestrations (centralized service compositions), adaptation of service choreographies (distributed service compositions), and integration of adaptation features in service composition processes. In all cases, we have developed prototypes to validate our approaches.

**Efficient Behavioral Adaptation for Service Orchestration.** Adaptation processes are either restrictive (pruning interaction sequences leading to deadlocks) or generative (enabling message reordering and generation when required). In our previous works we have developed a restrictive and generative adaptation approach [13]. Still, efficiency was an issue as pruning and adaptor behavioral reduction was performed after a raw (big) model had been generated. In [23] we have defined a more efficient adaptation technique, where both pruning and reduction are performed *while* the adaptor is computed. This approach is based on the encoding of service protocols (conversations) and value-passing adaptation contracts in the LOTOS process algebra and the use of on-the-fly model-checking and behavioral reduction techniques. Further, application to SOA has also been tackled by the automatic transformation of adaptor models into service orchestrations, which is achieved in two steps: (i) adaptor model filtering (to remove non-implementable message sequences) and, (ii) BPEL encoding.

**Behavioral and Semantic Adaptation for Service Choreographies.** While automatic adaptation is highly desirable for SOA where systems are composed from dynamically discovered services, component adaptation techniques do not support the semantic level and therefore require a mapping to be given by a designer to deal for example with message name mismatch between services. Service composition is usually achieved through orchestration. Still, distributed adaptation is an important issue in domains such as pervasive computing, due to the use of resource-limited devices and ad hoc networks, no centralized server being available to execute the orchestration/adaptor. In [24], we have tackled composition and adaptation from a complementary point of view, choreography. With reference to earlier works, adaptation is fully automatic thanks to an ontology that is used to generate implicit links between the choreography partners, which no longer requires an adaptation mapping to be given beforehand. Another interest of this approach is that it generates a distributed set of adaptors and avoids centralized adaptor implementations. More recently, we have studied this approach to overcome mismatch between client protocols and roles in auction protocol choreographies.

**Adaptation Features in the Service Composition Process.** Task-Oriented Computing (TOC) envisions a user-friendly world where user tasks would be achieved by the automatic assembly of resources available in the environment. Service-Oriented Computing (SOC) is a cornerstone towards the realization of this vision, through the abstraction of heterogeneous resources as semantic services described in terms of capacities and data flows, that is more abstractly than in our previous works based on protocols over operation calls. Still, this description level raises both vertical (between user task and service descriptions) and horizontal (between service data flows) mismatch issues. In [16] we have investigated a task-oriented service composition process based on graph planning techniques (graphplan) with two specific adaptation features: vertical mismatch is supported using task decomposition guidelines, horizontal mismatch is supported using ontologies. This technique generates service compositions (orchestrations) under the form of plans which are then translated into the BPEL service orchestration language.

Ongoing work is threefold. A first perspective is related to the development of *on-the-fly* algorithms for the behavioral reduction technique we have developed in 2007, and for the BPEL adaptor filtering presented in [23]. The objective is to gain in efficiency by applying both algorithms *while* computing the adaptors (either their centralized or decentralized forms) rather than after a raw model has been generated. A second perspective is related to the implementation of the distributed adaptors [24], using state-of-the art exogenous coordination languages. As a longer term perspective, we plan to study the application of automatic theorem provers and SAT-solvers to extend the adaptive planning technique [16] to services whose semantics would be described with pre- and post-condition based on first-order logic.

## 6.5. Privacy-Enhanced Service-Oriented Middleware

**Participants:** Valérie Issarny, Natallia Kokash, Roberto Speicys Cardoso, Pierre-Guillaume Raverdy.

Privacy is a major concern in today's information society. As computers are used to mediate a growing number of human activities, a greater amount of personal data is digitalized and can be easily transmitted, stored and analyzed. If, in the past, private information such as shopping habits, activities, preferences and whereabouts vanished with time, today this data can be either directly collected or deduced from other available data, and correlated to infer a number of an individual's personal details without one's awareness. Privacy protection becomes even more significant in pervasive computing systems, where the surrounding space incorporates computing devices that the user manipulates unconscious of their existence to execute daily activities. Pervasive information systems, thus, can collect a larger amount of personal data and amplify the possibilities for privacy invasions.

To effectively protect user's privacy, the infrastructure, middleware and application layers must cooperate, supported by a privacy legislation that enables legal measures against identified privacy invasions. In our work, we focus on the pervasive middleware, which must not only provide mechanisms to encourage the development of privacy-aware pervasive applications but also must be designed taking privacy into account. Particularly, we consider privacy issues that arise on service-oriented middleware both when applications interact with middleware services as well as on the middleware design and implementation. Our work focuses on the privacy risks introduced by the three basic functionalities of a service-oriented middleware for pervasive computing, namely *service discovery*, *service composition* and *service access*.

Regarding service discovery, we had already introduced and demonstrated a privacy-enhanced protocol for syntactic and semantic service discovery that reduces the trust requirements of service directories while enhancing privacy protection for both clients and service providers. Service descriptions are modified in such a way that a privacy-enhanced description can represent multiple original descriptions. Service directories are unable to relate a given privacy-enhanced service request or advertisement to the original request or advertisement, but are still able to find matches between privacy-enhanced service descriptions. As a result, service directories cannot recognize which services are announced or requested, which reduces the privacy risks posed by untrusted directories. Still concerning service discovery, we also had studied privacy issues that appear when performing multi-network service discovery. We proposed a series of mechanisms to (i) allow clients to control *where* service requests are forwarded, (ii) allow clients to control *how* service requests are forwarded and (iii) enhance middleware privacy-awareness when forwarding service discovery request and results to untrusted networks.

In 2008 we extended our work to handle privacy issues in service composition and service access. When composing services in open environments such as pervasive computing, hardly ever the is exact service description that the client requests available in the environment. To enable users to execute composite services in such scenarios, the pervasive middleware must provide a flexible mechanism for service composition that can compose available services at run-time to provide a composition functionally equivalent to the user-defined composite service. This feature, however, has an impact on the user privacy since the task partition specified on the user-defined abstract workflow may not be respected by the middleware-provided executable workflows, and service providers may be able to correlate data that the user wishes to keep disconnected. In [20] we propose a model based on an extension of Fuzzy Cognitive Maps that enable users to reason about the privacy impact of executing a service composition and to compare the consequences on privacy of running two functionally equivalent compositions featuring distinct partitions of tasks among service providers.

Finally, we addressed the issue of service access in hybrid mobile ad hoc networks in [19]. Pervasive computing environments combine structured and ad hoc networks using different technologies (WiFi, Bluetooth, Ethernet, GPRS, etc.). In such scenarios, it is common that packets are routed through untrusted user nodes and proper care must be taken to avoid that unauthorized nodes have access to message contents. Even though techniques based on public key encryption to solve that problem already exist and are widely deployed in environments such as the Internet, they are not suitable to ad hoc networks that cannot rely on an infrastructure for public key distribution. We propose to explore the diversity of paths that may exist between any two nodes in such networks to improve resistance to eavesdropping. Nodes can split a message into multiple parts and send each part through a different path in such a way that to have access to the whole message an attacker must possess a number of shares, making eavesdrop attacks harder to perform. The novelty of our approach is that we consider each node's software platform and each network's characteristics to compute paths that are resistant to attack to a given software layer (e.g. an operating system) or network technology (e.g. the encryption algorithm used by a network standard).

## 6.6. Data Sharing and Replication in Pervasive Networks

**Participants:** Amel Bennaceur, Pierre-Guillaume Raverdy, Pushpendra Singh, Valérie Issarny.

Nowadays, users rely on various sets of connected devices and services to interact with each other: home gateways, laptops, smartphones, UMPC, enterprise servers, Web/cloud computing services, etc. While for a long time interactions have been mostly Web-centric, users now routinely engage in user centric interactions such as content sharing and social networking through mobile devices. Still, while mobile devices have become very capable of communicating with other devices, accessing or exchanging content is far from being easy: communication costs are still high, communication protocols are not uniformly supported, communication links are not always reliable, etc. Another problem lies in the role usually assigned to mobile devices. Although mobile devices are the primary terminals for interaction, they have essentially focused on (i) acting as clients accessing services in the infrastructure (e.g., enablers in the IMS infrastructure for Telecoms networks, or Web services in the Internet), or (ii) manipulating content stored locally (e.g., multimedia content playback). So far, there has been little success in truly integrating these mobile devices in the pervasive computing environment (i.e., acting also as resource or service providers). Furthermore, the use of multiple devices, having large storage capabilities, has resulted in the scattering of content on a set of devices, which, ironically, is making access to all the content that one possesses a big challenge (Where is this file? Is it the latest version?). This problem becomes acute when it comes to impromptu collaboration (e.g., in business meetings, conferences, or on the road). Such collaborations often require exchange of content, and have to wait till the user gets physical access to the device over which the required content (or the correct/latest version) is currently stored. Delay in accessing the content often cause frustration and missed opportunities.

To answer these challenges, and better support interactions between mobile users, we are developing the iBICOOP middleware [17]. Our middleware addresses these challenges by targeting both fixed and mobile devices, leveraging their characteristics (e.g., always on and unlimited storage for home/enterprise servers, ad hoc communication link between mobile devices), and by leveraging the capabilities of all available networks (e.g., ad hoc networks, Internet, Telecoms infrastructure networks). It also relies on Web and Telecoms standards to promote interoperability.

The base architecture of the iBICOOP middleware consists of core modules on top of which we can develop applications that may arise in up-coming multi-device, multi-user world.

- *Communication Manager:* The iBICOOP Communication Manager is aimed to overcome the constraints of different network characteristics that different devices may have. The Communication Manager provides mechanisms to communicate over different available network interfaces of a device – Bluetooth, WiFi, Cellular – and also using different technologies e.g., Web services, HTTP/TCP sockets, ad-hoc mode. The communication between two devices is always secured with SSL.
- *Security Manager:* The iBICOOP Security Manager uses well-established techniques of cryptography and secure communication to provide necessary security. Presently, we are providing RSA, AES, DES, and Diffie-Hellman for generating keys that can be used for encryption/decryption of data for storing on device or sending over the net. The Security Manager is also responsible for access control on shared devices.
- *Partnership Manager:* The iBICOOP Partnership Manager provides device or user information in the form of *profiles*. Profiles are stored as XML files. This allows us to easily integrate our profiles with the IMS architecture using XDMS (XML Data Management Server). We have defined XML schema for profile. In the iBICOOP middleware, profiles are used by core modules to make a service available for user. To make our solution privacy-aware, the Partnership Manager module provides filters to control the information exchanged in a profile.
- *Naming & Discovery:* The iBICOOP's naming scheme is based on hierarchical names in the pattern of URI scheme[7]; We combine names of protocol, devices, services, email etc. to give a human-readable name to a resource. An example for a service is: "ibic://john@work.com:ipaqpda/axis2/services/exchange". For iBICOOP, we rely on Service Location Protocols to find nearby services on currently active network interfaces that support IP multicast. Setting up partnerships between users (bootstrapping a relationship) may also use the discovery service, and may happen after a multicast-based local discovery (e.g., over WiFi) for any Partnership Manager service, or by searching for such services on a public repository.
- *Local File Manager:* Besides normal file managing tasks, the iBICOOP Local File Manager gives user clear cues to the files that have been replicated across multiple devices or shared among different users by using different icons. Later it will provide extra functionalities for content-sharing and data management in emerging pervasive computing environments

We are currently implementing the iBICOOP middleware in Java, using IBM J9's JVM with CDC 1.1 for Windows mobile PDA and smartphones, as well as native CLDC/MIDP for other phones and smartphones. On the infrastructure side, services (e.g., Communication proxy service and Discovery service) are developed in Java 1.6, and deployed on Apache Tomcat and on the Alcatel 5350 IMS application server (for the IMS infrastructure). A number of core modules have already been implemented (Partnership Manager, Security Manager, and Communication Manager) on specific devices and are being ported. We plan to implement iBICOOP as a tight integration between services on the Internet, local networks, and in the IMS architecture thus providing a solution for both the Telecoms and Internet world.

We are also building two collaboration services on iBICOOP core modules: the *Replication Service* and the *Exchange & Sharing service*.

- *Replication Service:* The iBICOOP Replication Service keeps track of the files that are replicated over the user workspace. A workspace is the total space available on all the user devices. Users are made aware of conflicts if they try to synchronize with a copy that has been modified on another device. Users can add or remove files from replication service when they wish so. A multi-criteria algorithm is used to choose the best way to transfer data during replication.

[7] http://tools.ietf.org/html/rfc3986

- *Exchange & Sharing Service:* The Exchange and Sharing Service takes the abstraction to a higher level by allowing several users to interact and collaborate in various scenarios. No matter where data is located, users can transfer and share data between their workspaces. The aim of this service is to make long-term collaborations or impromptu exchanges an easier and secure task.

In iBICOOP, we are developing a middleware to allow ubiquitous access of user data from a multitude of devices with heterogeneous capabilities and running on different platforms. The services that we are building on top of iBICOOP aim to show the viability of iBICOOP as a standard-based platform for future advance services. Leveraging Telecoms and Internet world, integration with IMS architecture, is one of the salient feature of iBICOOP. The iBICOOP middleware offers a complete solution for end-user with regards to content-sharing and data management in emerging pervasive computing environments.

# 7. Other Grants and Activities

## 7.1. European Contracts and Grants

### *7.1.1. IST FP6 IP Amigo*

**Participants:** Sébastien Bianco, Nikolaos Georgantas, Valérie Issarny, Graham Thomson.

- **Name:** IST Amigo – *Ambient Intelligence for the networked home environment*
- **URL:** http://www.extra.research.philips.com/euprojects/amigo/index.htm
- **Related activities:** § 6.3
- **Period:** [September 2004 - February 2008]
- **Partners:** Philips Research Eindhoven (The Netherlands) – project coordinator, Philips Design - Philips Consumer Electronics (The Netherlands), Fagor (Spain), France Télécom (France), Fraunhofer IMS (Germany), Fraunhofer IPSI (Germany), Ikerlan (Spain), INRIA (URs Rocquencourt, Futurs, Loraine, Rhône Alpes), Italdesign Giugiaro (Italy), Knowledge (Greece), Microsoft (Germany), Telin (The Netherlands), ICCS (Greece), Telefónica I+D (Spain), University of Paderborn (Germany), VTT (Finland).

Home networking has already emerged in specific applications such as PC to PC communication and home entertainment systems, but its ability to really change people's lives is still dogged by complex installation procedures, the lack of interoperability between different manufacturer's equipment and the absence of compelling user services. By focusing on solving these key issues, the Amigo project aimed to overcome the obstacles to widespread acceptance of this new technology. The project has developed open, standardized, interoperable middleware and attractive user services, thus improving end-user usability. The project further showed the end-user usability and attractiveness of such a home system by creating and demonstrating prototype applications improving everyday life, addressing all vital user aspects: home care and safety, home information and entertainment, and extension of the home environment by means of ambiance sharing for advanced personal communication. The Amigo project further supports interoperability between equipment and services within the networked home environment by using standard technology when possible and by making the basic middleware (components and platform) and basic user services available as open source software together with architectural rules for everyone to use.

### *7.1.2. IST FP6 STREP PLASTIC*

**Participants:** Mauro Caporuscio, Valérie Issarny, Hassine Moungla, Pierre-Guillaume Raverdy.

- **Name:** IST PLASTIC – *Providing Lightweight and Adaptable Service Technology for Pervasive Information and Communication*
- **URL:** http://www.ist-plastic.org/

- **Related activities:** § 6.2
- **Period:** [February 2006 - September 2008]
- **Partners:** INRIA (UR Rocquencourt) – project coordinator, 4D Soft (Hungary), CNR (Italy), IBM (Belgium), Siemens Business Services (Germany), Telefónica I+D (Spain), UCL (United-Kingdom), Universitá di L'Aquila (Italy), Universitá della Svizzera Italiana (Switzerland), Virtual Trip (Greece), Pragmatica Technologies (Argentina).

The vision of PLASTIC is that users in the B3G era should be provided with a variety of application services exploiting the network's diversity and richness, without requiring systematic availability of an integrated network infrastructure. The success of the provided services then depends on the user perception of the delivered QoS. In particular, the network's diversity and richness must be made available and be exploitable at the application layer, where the delivered services can be most suitably adapted. This demands a comprehensive software engineering approach to the provisioning of services, which encompasses the full service life cycle, from development to validation, and from deployment to execution. The PLASTIC project aimed to offer a provisioning platform for software services deployed over B3G networks. The platform enables dynamic adaptation of services to the environment with respect to resource availability and delivered QoS, via a development paradigm based on Service Level Agreements and resource-aware programming. The middleware is service oriented, to enable integration and composition of heterogeneous software services from both infrastructure-based and ad hoc networks. The middleware further integrates key functions for supporting the management of adaptive services in the open wireless environment, dealing with resource awareness and dependability.

## 7.2. International Research Networks and Work Groups

### *7.2.1. ASA Associated Team*

- **Name:** ASA – *Adaptive SoftwAre*
- **Period:** [created 2007]
- **Participants:** Joint team with Universita dell'Aquila, Dipartimento di Informatica, Italy

The objective of the team is assisting the development of dynamic distributed software systems for next generation ubiquitous communication and computing infrastructures. Software in the near ubiquitous future (Softure) will need to cope with variability, as software systems get deployed on an increasingly large diversity of computing platforms and should further deliver applications ubiquitously. Heterogeneity of the underlying communication and computing infrastructures, mobility and continuously evolving requirements demand new software paradigms that span the entire life-cycle, from development to deployment and execution. Softure must be developed in a way that facilitates both its deployment over heterogeneous networks of heterogeneous nodes, and its interaction with end users, their environment and/or other existing systems, depending on the application domain. Moreover, Softure should be reliable and meet the user's performance requirements and needs. Last but not least, Softure should be dynamic so that the applications they implement can be provisioned ubiquitously, despite the high dynamics of the pervasive networking and computing environment. Looking at the software life cycle, one key issue in this domain appears to be the disappearance of a clear distinction between static and dynamic aspects. Indeed, the adaptability requirement imposed by ubiquity makes software become evolving in nature, therefore introducing a strong interaction between the development environment and the middleware one. The goal of the ASA associated team is to research design and programming techniques and innovative middleware models that can be profitably integrated to support this new generation of software systems.

### *7.2.2. ESF Scientific Programme MiNEMA*

- **Name:** ESF Scientific Programme – *Middleware for Network Eccentric and Mobile Applications*
- **URL:** http://www.minema.di.fc.ul.pt/index.html

- **Period:** [September 2003 - August 2008]
- **Steering Committee:** University Klagenfurt (Austria), KU Leuven (Belgium), University of Cyprus (Cyprus), Aarhus University (Denmark), University of Helsinki (Finland), University of Ulm (Germany), TCD (Ireland), University of Lisboa (Portugal), CTH (Sweden), EPFL (Switzerland), Lancaster University (UK).

MiNEMA is a European Science Foundation (ESF) Scientific Program aiming to bring together European groups from different communities working on middleware for mobile environment. The program intends to foster the definition and implementation of widely recognized middleware abstractions for new and emerging mobile applications. The program includes the following planned activities:

- Short term visit exchanges among the program participants (PhD students).
- Organization of a "closed" workshop for program participants, to allow the dissemination of early research results and experiences.
- Sponsoring of workshops and conferences in the area of MiNEMA.
- Organization of a summer school on the subjects covered by the program.

### *7.2.3. ERCIM WG RISE*

- **Name:** ERCIM Working Group – *Rapid Integration of Software Engineering Techniques*
- **URL:** http://rise.uni.lu/tiki/tiki-index.php
- **Period:** [Created 2004]
- **Participants:** CCLRC (UK), CNR (Italy), CWI (The Netherlands), FNR (Luxembourg), FORTH (Greece), Fraunhofer FOKUS & IPSI (Germany), INRIA (UR Rocquencourt), LIRMM (France), NTNU (Norway), SARIT (Switzerland), SICS (Sweden), SpaRCIM (Spain), SZTAKI (Hungary), University of Newcastle (UK), VTT (Finland).

The main aim of the RISE working group is to conduct research on providing new, integrated and practical software engineering approaches that are part of a methodological framework and that apply to new and evolving applications, technologies and systems. In order not to consider all the scope of software engineering, the RISE working group focuses on the following sub domains: Software/Systems Architectures, Reuse, Testing, Model Transformation/Model Driven Engineering, Requirement Engineering, Lightweight formal methods, and CASE tools.

The RISE working group limits also its researches to specific application domains for the problems and solutions it proposes. The starting application domains proposed are: Web Systems, Mobility in Communication Systems, High Availability Systems, and Embedded Systems.

### *7.2.4. ERCIM WG SESAMI*

- **Name:** ERCIM Working Group – *Smart Environments and Systems for Ambient Intelligence*
- **URL:** http://www.ics.forth.gr/sesami/
- **Period:** [Created 2006]
- **Participants:** LORIA (France), INRIA (UR Rocquencourt), CNR (Italy), University of Luxemburg (Luxembourg), VTT (Finland), SpaRCIM (Spain), Fraunhofer FOKUS & IPSI (Germany), University College Dublin (UK), CWI (The Netherlands), Middlesex University (UK), KU Leuven (Belgium), IMAG (France), University of Linz (Austria), University of Graz (Austria), University of Thessaly (Greece), University of Salzburg (Austria), University of Kiel (Germany), University of Kaiserslauten (Germany), University of Munich (Germany).

Ambient Intelligence represents a vision of the (not too far) future where "intelligent" or "smart" environments and systems react in an attentive, adaptive, and active (sometimes even proactive) way to the presence and activities of humans and objects in order to provide intelligent/smart services to the inhabitants of these environments.

Ambient Intelligence technologies integrate sensing capabilities, processing power, reasoning mechanisms, networking facilities, applications and services, digital content, and actuating capabilities distributed in the surrounding environment. While a wide variety of different technologies is involved, the goal of Ambient Intelligence is to hide their presence from users, by providing implicit, unobtrusive interaction paradigms. People and their social situations, ranging from individuals to groups, be them work groups, families or friends and their corresponding environments (office buildings, homes, public spaces, etc) are at the center of the design considerations.

The ERCIM Working Group SESAMI aims to facilitate the continued collaboration of researchers and practitioners working on the design, implementation and evaluation of Ambient Intelligence systems and applications, on the grounds of ongoing, and potentially cross-domain, basic and applied, research and development. In this context, SESAMI will pursue novel insights on designing, implementing, managing and maintaining smart computational environments of any scale, in order to effectively enhance and go beyond traditional support of human activities for any given situation, context, role, mission, and task.

### *7.2.5. ERCIM WG STM*

- **Name:** ERCIM Working Group – *Security and Trust Management*
- **URL:** http://www.iit.cnr.it/STM-WG/
- **Period:** [Created 2005]
- **Participants:** British Telecom, CLRC (UK), CNR (Italy), CETIC (Belgium), CWI (The Netherlands), DTU (Denmark), FORTH-ICS (Greece), FNR (Luxembourg), Fraunhofer-SIT (Germany), HP, IBM Research, INRIA (URs Rocquencourt & Sophia Antipolis), IUC (Ireland), L3S (Germany), Marasyk University (Czech Republic), Microsoft EMIC (Germany), NTNU (Norway), Politecnico Torino (Italy), SAP (Germany), SARIT (Switzerland), SICS (Sweden), Siemens Corporate Technology, SparCIM (Spain), SZTAKI (Hungary), VTT (Finland), Eindhoven University of Technology (The Netherlands), University of Milan (Italy), University of Roma Tor Vergata (Italy), University of Trento (Italy), University of Twente (The Netherlands), VCPC, W3C.

The pervasive nature of the emerging Information and Communication Technologies (ICT) expands the well known current security problems on ICT, due to the increased possibilities of exploiting existing vulnerabilities and creating new threats. On the other hand, it poses new problems in terms of possible attack scenarios, threats, menaces and damages. Moreover, the increased virtual and physical mobility of the users enhances their interaction possibilities. Thus, there is a demand for a reliable establishment of trust relationships among the users. Privacy is also a main concern in the current ambient intelligence paradigm: everywhere there are devices interacting with users and information about the users is possibly being gathered by the devices at anytime. All these problems are perceived at different levels of concern by users, technology producers, scientific and governance communities.

This ERCIM Working Group aims at focusing the research of the ERCIM institutions on a series of activities (*e.g.*, projects and workshops) for fostering the European research and development on security, trust and privacy in ICT. These will be among the main issues of current and future research efforts for "security" in a broad sense in Europe (http://www.cordis.lu/security/).

## 7.3. National Contacts and Grants

### *7.3.1. EXOTICUS: Etude et eXpérimentation des Outils & Technologies IMS Compatibles avec les USages*

**Participants:** Mauro Caporuscio, Valérie Issarny, Pierre-Guillaume Raverdy, Amel Bennaceur, Pushpendra Singh.

- **Name:** EXOTICUS – *Etude et eXpérimentation des Outils & Technologies IMS Compatibles avec les USages*

- **Related activities:** § 6.6
- **Period:** [November 2007 – October 2009]
- **Partners:** Alcatel Lucent - coordinator, Legos, PragmaDev, Archos, Citypassenger, Deveryware, Transatel, ENST, GET/INT, INRIA (CRI Paris-Rocquencourt)

Introduced by the 3GPP standardization forum in relation with 3G, IMS (IP Multimedia Subsystem) is a key element for telecom operators trying to increase the status of their service provider activities, in a highly concurrent marketplace. The incoming IMS should enable the interconnexion of fixed and mobile access networks around a same IP core, and making an open service platform available. The objective of the EXOTICUS project is to tackle the technological locks related to this new architecture, by contributing to:

- bridge the weakness of the norm about processes for the service composition and integration,
- foster the service creation dynamics thanks to new innovative service primitives,
- accelerate the development cycles, experiment new services and assess their use.

This research activities will bring fuel, all along the project, to the experiment platform that is deployed in southern Ile de France, and which is available for both professional (especially from the telecom and automotive sectors) and public population.

### *7.3.2. PERSO: PERvasive Service cOmposition*

**Participants:** Manel Fredj, Nikolaos Georgantas, Pascal Poizat.

- **Name:** PERSO – *PERvasive Service cOmposition*
- **Related activities:** § 6.4
- **Period:** [November 2007 – October 2010]
- **Partners:** INRIA (CRI Paris-Rocquencourt) - coordinator, University Paris Dauphine - LAMSADE, University of Evry - IBISC.

The objectives of the project comprise the study, analysis and elaboration of a comprehensive approach to service composition for pervasive environments, supporting three description levels for service interfaces that enrich the standard description level for service interfaces (signatures): behavioral level, non-functional level and semantic level. We specifically aim at supporting these three levels and their integration towards a thorough approach to service composition. Two axes are further identified. The first axis concerns the foundations of service composition in terms of underlying formal models and related algorithms for service interface specification, service discovery, and service composition, including adaptation. The second axis concerns the application of the elaborated first axis outcomes to the runtime pervasive environment, tackling issues such as: efficiency and performance of algorithms for the interactive pervasive environment; resource consumption on resource-constrained portable devices; monitoring mechanisms for detecting change of conditions and for triggering composite service reconfiguration; runtime mechanisms for ensuring QoS in the face of change.

### *7.3.3. SemEUsE: SEMantiquE pour bUS de sErvice*

**Participants:** Nebil Ben Mabrouk, Nikolaos Georgantas, Valérie Issarny, Mauro Caporuscio.

- **Name:** SEMEUSE – *SEMantiquE pour bUS de sErvice*
- **Related activities:** § 6.3
- **Period:** [December 2007 – May 2010]
- **Partners:** Thales - coordinator, France Télécom R&D, EBM WebSourcing, Université Pierre et Marie Curie - LIP6/MoVe, INSA Lyon, INRIA (CRI Paris-Rocquencourt), ObjectWeb, GET/INT.

The aim of the SemEUsE project is to study a semantic based service infrastructure that will provide the foundational services required for service-oriented applications to exchange information in a ubiquitous, reliable environment. The combination of an emerging semantic service infrastructure and associated engineering techniques will make it possible to produce flexible, mission-critical, software-based service applications that are dependable and manageable, and to provide high levels, business-focused, guaranteed end-to-end quality-of-services for all users.

# 8. Dissemination

## 8.1. Involvement within the Scientific Community

### *8.1.1. Program Committees*

- Mauro Caporuscio is PC Member of CAMPUS 2008 and AdhocAmC'08;
- Mauro Caporuscio has been co-organizer and PC Co-Chair of the 1st International Workshop on Automated engineeRing of Autonomous and run-tiMe evolvIng Systems (ARAMIS 2008) and the 1st International Workshop on Context-aware Adaptation Mechanisms for Pervasive and Ubiquitous Services (CAMPUS 2008);
- Valérie Issarny and Nikolaos Georgantas are organizers and PC chairs of AdHocAmC'08: 1st International Workshop on Ad Hoc Ambient Computing at the 7th International Conference on Ad Hoc Networks and Wireless (AdHoc-NoW'08);
- Nikolaos Georgantas is PC member of the Middleware'08, DAIS'08, ICSOFT'08 and PerSys'08 international conferences;
- Nikolaos Georgantas is PC member of the IUI4AAL, RoSOC-M'08, SIPE'08, ARAMIS'08, MPAC'08 and MW4SOC'08 international workshops;
- Valérie Issarny is the PC co-chair of ACM/IFIP/USENIX Middleware 2008, 9th International Middleware Conference, December 1-5, 2008, Leuven, Belgium, and PC chair of ESEC/ACM SIGSOFT FSE'2009, the 7th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, August 24-28, 2008, Amsterdam, The Netherlands;
- Valérie Issarny is PC member of Autonomics'08, FASE'08, ICSE'09, IFIPTM'08 & 09, Middleware'09, WICSA'08 & 09;
- Pascal Poizat is co-organizer and PC co-chair of FOCLASA'08: 7th International Workshop on Foundations of Coordination Languages and Software Architectures, at the 35th International Colloquium on Automata, Languages and Programming (ICALP'08);
- Pascal Poizat is PC member of the IEEE ICEBE'08 International Conference and the FACS'08, SASE'08 and WASELF-*'08 International Workshops.

### *8.1.2. Other Activities*

- Mauro Caporuscio has been Publicity Chair of 23rd IEEE/ACM International Conference on Automated Software Engineering;
- Nikolaos Georgantas is member of the PhD monitoring committee at INRIA Paris-Rocquencourt;
- Valérie Issarny is associate editor of ACM Computing Surveys;
- Valérie Issarny chairs the executive committee of the European AIR&D consortium (www.air-d.org);
- Valérie Issarny is member of the INRETS scientific council & "Commission d'évaluation des chercheurs";

- Valérie Issarny is member of the LIESP scientific council;
- Valérie Issary is member of the GDR GPL scientific council ;
- Valérie Issarny is member of the evaluation committee of the ANR ARPEGE call on "Systèmes Embarqués et Grandes Infrastructures" in 2008 & 2009, and of the evaluation committee of the ANR "Programme Blanc" and "Programme Jeunes Chercheuses et Jeunes Chercheurs" for the 2009 call;
- Valérie Issarny is member of "jury INRIA CR2" of INRIA Lille-Nord Europe (2008);
- Animesh Pathak was the Production Chair of HiPC 2008, and Web Publicity Chair of DCOSS 2008;
- Pascal Poizat is co-editor and member of the review committee of the special issue on the 6th International Workshop on Foundations of Coordination Languages and Software Architectures of the journal Science of Computer Programming;
- Pascal Poizat is member of the recruiting commission for Computer Science Assistant Professors in University of Evry, France;
- Pascal Poizat was member of the council of laboratory of the IBISC FRE 3190 CNRS, Evry, France.

## 8.2. Teaching

- Nebil Ben Mabrouk gives a course on Distributed Objects Architectures (laboratory). Final year of the five-year computer engineering degree at the Institut des Sciences et Techniques des Yvelines of the University of Versailles Saint-Quentin en Yvelines;
- Manel Fredj gives a course on Middleware Architectures (laboratory). Final year of the five-year computer engineering degree at the Ecole Supérieure d'Ingénierie Léonard de Vinci of the Pôle Universitaire Léonard de Vinci;
- Nikolaos Georgantas gives a course on Middleware Architectures (lectures). Final year of the five year computer engineering degree at the Ecole Supérieure d'Ingénierie Léonard de Vinci of the Pôle Universitaire Léonard de Vinci;
- Nikolaos Georgantas gives a course on Distributed Objects Architectures (lectures). Final year of the five year computer engineering degree at the Institut des Sciences et Techniques des Yvelines of the University of Versailles Saint-Quentin en Yvelines;
- Valérie Issarny gives a course on Software Architectures for Distributed Systems (lectures), as part of the SAL course of the Master 2 COSY of the University of Versailles Saint-Quentin en Yvelines.

## 8.3. Internships

During year 2008, members of the ARLES project-team supervised the work of the following student interns:

- Olfa Ben Hadj Alaya, *Middleware pour la substitution dynamique des services Web dans les environnements diffus*, Projet de fin d'études d'ingénieur, spécialité génie logiciel , *Institut National des Sciences Appliquées et de Technologie* (INSAT), Tunis, Tunisie.
- Nikolaos Papanikos, *Exploring path diversity in multi-radio networks*, University of Ioannina, Ioannina, Greece.

# 9. Bibliography

## Major publications by the team in recent years

[1] S. BEN MOKHTAR, N. GEORGANTAS, V. ISSARNY. *COCOA: COnversation-based Service Composition in PervAsive Computing Environments with QoS Support*, in "Journal of Systems and Software, Special Issue on ICPS'06", vol. 80, n$^{o}$ 12, 2007, p. 1941–1955.

[2] S. BEN MOKHTAR, D. PREUVENEERS, N. GEORGANTAS, V. ISSARNY, Y. BERBERS. *EASY: Efficient SemAntic Service DiscoverY in Pervasive Computing Environments with QoS and Context Support*, in "Journal of Systems and Software, Special Issue on Web Services Modelling and Testing", vol. 81, n° 5, 2008, p. 785-808.

[3] Y.-D. BROMBERG, V. ISSARNY. *INDISS: Interoperable Discovery System for Networked Services*, in "Proceedings of ACM/IFIP/USENIX 6th International Middleware Conference (Middleware'2005)", 2005.

[4] M. CAPORUSCIO, P.-G. RAVERDY, H. MOUNGLA, V. ISSARNY. *ubiSOAP: A Service Oriented Middleware for Seamless Networking*, in "Proceedings of 6th International Conference on Service Oriented Computing (ICSOC'08)", 2008.

[5] D. CHARLET, V. ISSARNY, R. CHIBOUT. *Energy-efficient middleware-layer multi-radio networking: An assessment in the area of service discovery*, in "Comput. Netw.", vol. 52, n° 1, 2008, p. 4–24.

[6] V. ISSARNY, M. CAPORUSCIO, N. GEORGANTAS. *A Perspective on the Future of Middleware-based Software Engineering*, in "FOSE '07: 2007 Future of Software Engineering, Washington, DC, USA", IEEE Computer Society, 2007, p. 244–258.

[7] V. ISSARNY, D. SACCHETTI, F. TARTANOGLU, F. SAILHAN, R. CHIBOUT, N. LEVY, A. TALAMONA. *Developing Ambient Intelligence Systems: A Solution based on Web Services*, in "Automated Software Engg.", vol. 12, n° 1, 2005, p. 101–137.

[8] P.-G. RAVERDY, V. ISSARNY, R. CHIBOUT, A. DE LA CHAPELLE. *A Multi-Protocol Approach to Service Discovery and Access in Pervasive Environments*, in "Proceedings of MOBIQUITOUS - The 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services", 2006.

[9] F. SAILHAN, V. ISSARNY. *Energy-aware Web Caching over Hybrid Networks*, in "Handbook of Mobile Computing", CRC Press, 2004.

[10] F. SAILHAN, V. ISSARNY. *Scalable Service Discovery for MANET*, in "Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom'2005)", 2005.

# Year Publications

## Articles in International Peer-Reviewed Journal

[11] D. ATHANASOPOULOS, A. ZARRAS, V. ISSARNY, E. PITOURA, P. VASSILIADIS. *CoWSAMI: Interface-Aware Context Gathering in Ambient Intelligence Environments*, in "Pervasive and Mobile Computing Journal, Elsevier", vol. 4, n° 3, June 2008.

[12] S. BEN MOKHTAR, D. PREUVENEERS, N. GEORGANTAS, V. ISSARNY, Y. BERBERS. *EASY: Efficient SemAntic Service DiscoverY in Pervasive Computing Environments with QoS and Context Support*, in "Journal of System and Software", vol. 81, n° 5, May 2008.

[13] C. CANAL, P. POIZAT, G. SALAÜN. *Model-based Adaptation of Behavioural Mismatching Components*, in "IEEE Transactions on Software Engineering", vol. 34, n° 4, 2008, p. 546–563.

[14] D. CHARLET, V. ISSARNY, R. CHIBOUT. *Energy-efficient Middleware-layer Multi-radio Networking: An Assessment in the Area of Service Discovery*, in "Computer Networks", vol. 52, n$^{o}$ 1, January 2008.

[15] A. ZARRAS, P. VASSILIADIS, V. ISSARNY. *Modeling and analyzing reliable service-oriented processes*, in "International Journal on Business Process Integration and Management.", to appear, 2009.

### International Peer-Reviewed Conference/Proceedings

[16] S. BEAUCHE, P. POIZAT. *Automated Service Composition with Adaptive Planning*, in "Proceedings of the International Conference on Service Oriented Computing (ICSOC 08)", Lecture Notes in Computer Science, vol. 5364, Springer, 2008, p. 530–537.

[17] A. BENNACEUR, P. SINGH, P.-G. RAVERDY, V. ISSARNY. *The iBICOOP middleware: Enablers and Services for Emerging Pervasive Computing Environments*, in "Proceedings of IEEE PerWare'09", to appear, 2009.

[18] M. CAPORUSCIO, P.-G. RAVERDY, H. MOUNGLA, V. ISSARNY. *ubiSOAP: A Service Oriented Middleware for Seamless Networking*, in "Proceedings of 6th International Conference on Service Oriented Computing (ICSOC'08)", 2008.

[19] R. S. CARDOSO, M. CAPORUSCIO. *Exploring Multi-path Communication in Hybrid Wireless Ad Hoc Networks*, in "Proceedings of 1st International Workshop on Ad-hoc Ambient Computing, Sophia Antipolis, France", September 2008.

[20] R. S. CARDOSO, V. ISSARNY. *A Model for Reasoning About the Privacy Impact of Composite Service Execution in Pervasive Computing*, in "Proceedings of IFIPTM 2008 Joint iTrust and PST Conferences on Privacy, Trust Management and Security, Trondheim, Norway", June 2008.

[21] M. FREDJ, N. GEORGANTAS, V. ISSARNY, A. ZARRAS. *Dynamic Service Substitution in Service-Oriented Architectures*, in "Proceedings of the IEEE Congress on Services (Services 2008-SCC 2008), SOA Industry Summit, Hawaii, USA", July 2008.

[22] N. KOKASH, R. S. CARDOSO, V. ISSARNY, P.-G. RAVERDY. *A Flexible QoS-aware Routing Protocol for Infrastructure-less B3G Networks*, in "Proceedings of ACM SAC - The 24th Annual ACM Symposium on Applied Computing - Track on Mobile Computing and Applications", to appear, 2009.

[23] R. MATEESCU, P. POIZAT, G. SALAÜN. *Adaptation of Service Protocols using Process Algebra and On-the-Fly Reduction Techniques*, in "Proceedings of the International Conference on Service Oriented Computing (ICSOC 08)", Lecture Notes in Computer Science, vol. 5364, Springer, 2008, p. 84–99.

[24] T. MELLITI, P. POIZAT, S. BEN MOKHTAR. *Distributed Behavioural Adaptation for the Automatic Composition of Semantic Services*, in "Proceedings of the International Conference on Fundamental Approaches to Software Engineering (FASE 08)", Lecture Notes in Computer Science, vol. 4961, Springer, 2008, p. 146–162.

### Scientific Books (or Scientific Book chapters)

[25] S. BEN MOKHTAR, N. GEORGANTAS, V. ISSARNY, P.-G. RAVERDY, M. AUTILI. *Service Discovery in Pervasive Computing Environments*, in "At Your Service: Service-Oriented Computing from an EU Perspective", ISBN 978-0-262-04253-6, The MIT Press, Cambridge/Massachusetts, London/England, 2009.

[26] N. GEORGANTAS, V. ISSARNY, J. KANTOROVITCH, J. KALAOJA, I. ROUSSAKI, I. PAPAIOANNOU, D. TSESMETZIS. *Amigo: Interoperable semantic services for the smart home environment*, in "At Your Service: Service-Oriented Computing from an EU Perspective", ISBN 978-0-262-04253-6, The MIT Press, Cambridge/Massachusetts, London/England, 2009.

[27] F. SAILHAN, J. BOURGEOIS, V. ISSARNY. *A Security Supervision System for Hybrid Networks*, in "Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing", 2008, p. 137-149.

### Books or Proceedings Editing

[28] V. ISSARNY, R. SCHANTS (editors). *Proceedings of the ACM/IFIP/USENIX 9th International Middleware Conference*, vol. 5346, LNCS, December 2008.