Research-Team ArchWare

Software Architecture

IRISA Site: Vannes

Activity Report

2014

1 Team

Head of the team

Flavio Oquendo, Full Professor, Université de Bretagne-Sud

Administrative assistant

Sylviane Boisadan, BIATSS, Université de Bretagne-Sud

Université de Bretagne-Sud

Isabelle Borne, Full Professor
Jérémy Buisson, Assistant Professor
Régis Fleurquin, Associate Professor, HDR
Elena Leroux, Assistant Professor
Salah Sadou, Associate Professor, HDR
Gersan Moguérou, Research Engineer

Visitors

Thais Batista, Associate Professor, University of Rio Grande do Norte - UFRN (Brazil)
Flávia Delicato, Associate Professor, University of Rio de Janeiro - UFRJ (Brazil)
Jair Leite, Associate Professor, University of Rio Grande do Norte - UFRN (Brazil)
Djamel Meslati, Associate Professor, University of Annaba (Algeria)
Elisa Nakagawa, Associate Professor, University of Sao Paulo - USP (Brazil)
Paulo Pires, Associate Professor, University of Rio de Janeiro - UFRJ (Brazil)
Danny Weyns, Full Professor, University of Linnaeus (Sweden)

PhD students

Youcef Bouziane, Cotutelle grant, since September 2014
Everton Cavalcante, CsF-CNPq grant, since October 2013
Imane Cherfa, Cotutelle grant, since September 2014
Franck Petitdemange, MESR grant, since September 2014
Tahar Gherbi, Cotutelle grant, defended his thesis on June 2014
Marcelo Gonçalves, CsF-CNPq grant, since October 2013
Milena Guessi, FAPESP grant, since September 2014
Salma Hamza, ARED grant, defended her thesis on December 2014
Davy Hélard, CIFRE grant with MGDIS, since September 2012
Soraya Mesli, CIFRE grant with SEGULA, since November 2013
Lucas Oliveira, CsF-CNPq grant, since October 2012

Abderrhamane Seriai, ARED grant with University of Montreal, defended his thesis on January 2015
Minh Tu Ton That, MESR grant, defended his thesis on October 2014

2 Overall Objectives

2.1 Overview

The ArchWare Research Team addresses the scientific and technological challenges raised by architecting complex software-intensive systems. Beyond large-scale distributed systems, it addresses an emergent class of evolving software-intensive systems that is increasingly shaping the future of our software-reliant world, the so-called Systems-of-Systems (SoS).

Since the dawn of computing, the complexity of software and the complexity of systems reliant on software have grown at a staggering rate. In particular, software-intensive systems have been rapidly evolved from being stand-alone systems in the past, to be part of networked systems in the present, to increasingly become systems-of-systems in the coming future.

De facto, systems have been independently developed, operated, managed, and evolved. Progressively, networks made communication and coordination possible among these autonomous systems, yielding a new kind of complex system, i.e. a system that is itself composed of systems. These systems-of-systems are evolutionary developed from systems to achieve missions not possible by each constituent system alone.

Different aspects of our lives and livelihoods have become overly dependent on some sort of software-intensive system-of-systems. This is the case of systems-of-systems found in different areas as diverse as aeronautics, automotive, energy, healthcare, manufacturing, and transportation; and applications that addresses societal needs as e.g. in environmental monitoring, distributed energy grids, emergency coordination, global traffic control, and smart cities.

Moreover, emergent platforms such as the Internet of Things or the Internet of Everything and emergent classes of systems-of-systems such as Cyber-Physical Systems are accelerating the need of constructing rigorous foundations, languages, and tools for supporting the architecture and engineering of resilient systems-of-systems.

Complexity is intrinsically associated to systems-of-systems by its very nature that implies emergent behavior: in systems-of-systems, missions are achieved through emergent behavior drawn from the interaction among constituent systems. Hence, complexity poses the need for separation of concerns between architecture and engineering: (i) architecture focuses on reasoning about interactions of parts and their emergent properties; (ii) engineering focuses on designing and constructing such parts and integrating them as architected.

Definitely, Software Architecture forms the backbone for taming the complexity of critical software-intensive systems, in particular in the case of systems-of-systems, where architecture descriptions provide the framework for designing, constructing, and dynamically evolving such complex systems, in particular when they operate in unpredictable open-world environments.

Therefore, the endeavour of constructing critical systems evolved from engineering complicated systems in the last century, to architecting critical SoSs in this century. Critical SoSs, by their very nature, have intrinsic properties that are hard to address.

Furthermore, the upcoming generation of critical SoSs will operate in environments that are open in the sense of that they are only partially known at design-time. These open-world critical systems-of-systems, in opposite to current closed-world systems, will run on pervasive devices and networks providing services that are dynamically discovered and used to deliver more complex services, which themselves can be part of yet more complex services and so on. Furthermore, they will often operate in unpredictable environments.

Definitely, the unique characteristics of SoS raise a grand research challenge for the future of software-reliant systems in our industry and society due to its simultaneous intrinsic features, which are:

1. *Operational independence:* the participating systems not only can operate independently, they do operate independently. Hence, the challenge is to architect and construct SoS in a way that enables its operations (acting to fulfil its own mission) without violating the independence of its constituent systems that are autonomous, acting to fulfil their own missions.

2. *Managerial independence:* the participating systems are managed independently, and may decide to evolve in ways that were not foreseen when they were originally composed. Hence, the challenge is to architect and construct a SoS in a way that it is able to evolve itself to cope with independent decisions taken by the constituent systems and hence be able to continually fulfil its own mission.

3. *Distribution of constituent systems:* the participating systems are physically decoupled. Hence, the challenge is to architect and construct the SoS in a way that matches the loose-coupled nature of these systems.

4. *Evolutionary development:* as a consequence of the independence of the constituent systems, a SoS as a whole may evolve over time to respond to changing characteristics of its environment, constituent systems or of its own mission. Hence, the challenge is to architect and construct SoS in a way that it is able to evolve itself to cope with these three kinds of evolution.

5. *Emergent behaviours:* from the collaboration of the participating systems may emerge new behaviours. Furthermore, these behaviours may be ephemeral because the systems composing the SoS evolve independently, which may impact the availability of these behaviours. Hence, the challenge is to architect and construct a SoS in a way that emergent behaviours and their subsequent evolution can be discovered and controlled.

In the case of an open-world environment, one can add the following characteristics:

1. *Unpredictable environment:* the environment in which the open-world SoS operates is only partially known at design-time, i.e. it is too unpredictable to be summarised within a fixed set of specifications, and thereby there will inevitably be novel situations to deal with at run-time. Hence, the challenge is to architect and construct such a system in a way that it can dynamically accommodate to new situations while acting to fulfil its own mission.

2. *Unpredictable constituents:* the participating systems are only partially known at design-time. Hence, the challenge is to architect and construct an open-world SoS in a way that constituent systems are dynamically discovered, composed, operated, and evolved in a continuous way at run-time, in particular for achieving its own mission.

3. *Long-lasting:* as an open-world SoS is by nature a long-lasting system, re-architecting must be carried out dynamically. Hence, the challenge is to evolutionarily re-architects and evolves its construction without interrupting it.

The importance of developing novel theories and technologies for architecting and engineering SoSs is highlighted in several roadmaps targeting year 2020 and beyond.

In France, it is explicitly targeted in the report prepared by the French Ministry of Economy as one of the key technologies for the period 2015-2025 (étude prospective sur les technologies clés 2015-2025, Direction Générale de la Compétitivité, de l'Industrie et des Services du Ministére de l'Economie).

In Europe, SoSs are explicitly targeted in the studies developed by the initiative of the European Commission, i.e. Directions in Systems-of-Systems Engineering, and different Networks of Excellence (e.g. HiPEAC) and European Technological Platforms (e.g. ARTEMIS, NESSI) for preparing the Horizon 2020 European Framework Programme. In 2014, two roadmaps for systems-of-systems having been proposed, supported by the European Commission, issued from the CSAs ROAD2SoS (Development of Strategic Research and Engineering Roadmaps in Systems-of-Systems) and T-Area-SoS (Trans-Atlantic Research and Education Agenda in Systems-of-Systems).

All these key actions and the roadmaps for 2015-2020-2025 show the importance of progressing from the current situation, where SoSs are basically developed in ad-hoc way, to a scientific approach providing rigorous theories and technologies for mastering the complexity of software-intensive systems-of-systems.

Overall, the long-term research challenge raised by SoSs calls for a novel paradigm and novel trustful approaches for architecting, analyzing, constructing, and assuring the continuous correctness of systems-of-systems, often deployed in unpredictable environments, taking into account all together their intrinsic characteristics.

Actually, in the literature, SoSs are classified according to four categories:

1. *Directed SoS:* a SoS that is centrally managed and which constituent systems have been especially developed or acquired to fit specific purposes in the SoS - they operate under tight subordination;

2. *Acknowledged SoS:* a SoS that is centrally managed and that operates under loose subordination - the constituent systems retain their operational independence;

3. *Collaborative SoS:* a SoS in which there is no central management and constituent systems voluntarily agree to fulfil central purposes;

4. *Virtual SoS:* a SoS in which there is no central authority or centrally agreed purpose.

Regarding the state-of-the-art, software-intensive system-of-systems is an emergent domain in the research community. The systematic mapping of the literature shows that 75% of the

publications related to the architecture of systems-of-systems have been published in the last 5 years and 90% in the last 10 years. Furthermore, most of these publications raise open-issues after having experimented existing approaches for architecting systems-of-systems.

Our last systematic literature review searching all key bibliographic databases relevant to computer science, software and systems engineering, looking for publications from academia and experience reports from industry shows that, today, proposed approaches only support the architecture and construction of SoS matching the core characteristics, basically directed and acknowledged SoSs limited to non-critical systems. Therefore, achieving the targeted breakthrough will be a major advance in the state-of-the-art by delivering a conceptual, theoretical, and technological foundation for architecting and constructing the new generation of critical SoSs, i.e. collaborative and virtual SoSs. For addressing the scientific challenge raised for architecting SoS, the targeted breakthrough for the ArchWare Research Team is to conceive sound foundations and a novel holistic approach for architecting open-world critical software-intensive systems-of-systems, encompassing:

1. suitable architectural abstractions for formulating the architecture and re-architecture of SoS;

2. an appropriate formalism and its underlying computational model to rigorously specify the architecture and re-architecture of SoS;

3. the supporting mechanisms to construct and manage SoSs driven by architecture descriptions, while resiliently enforcing their correctness, effectiveness, and efficiency;

4. the supporting mechanisms to evolve SoSs driven by architecture descriptions, while resiliently enforcing their correctness, effectiveness, and efficiency throughout the evolution.

The research approach we adopt in the ArchWare Research Team for developing the expected breakthrough is based on well-principled design decisions:

1. to conceive architecture description, analysis, transformation, and evolution languages based on suitable SoS architectural abstractions;

2. to formally ground these SoS-specific architecture languages on well-established concurrent constraint process calculi and associated logics;

3. to conceptually and technologically ground the construction and management of SoSs on architecture descriptions defined by models-at-runtime in the service-oriented computing style;

4. to architecturally and technologically ground the dynamic evolution of SoS architectures around the concept of "anytime architecture".

2.2 Key Issues

As stated, an SoS can be perceived as a composition of systems in which its constituents, i.e. themselves systems, are separately discovered, selected, and composed possibly at run-time to form a more complex system, the SoS. Hence, four points are at the core of our approach:

- SoS architectures as dynamic compositions of systems: an SoS depends on composed systems and their interactions to undertake capabilities. Composition is thereby more challenging in SoS as systems being combined are active. This challenge beyond the state-of-the-art will be overcome in our approach by the development of SoS-specific composition mechanisms that are explicit, formally defined, and operate on active architectural models at run-time.

- Analysis of SoS architectures: SoS architecture descriptions, by their formal nature, will support automated analysis with a view in particular to evaluating and predicting non-functional qualities of the modelled SoS. In our approach, we will make a significant progress beyond the state-of-the-art of SoS analysis by developing techniques and tools for the architecture-centric model-based analysis of SoS. It will support verification of different sorts of properties and interleaving of these properties, including structural (e.g. connectivity, topology), behavioural (e.g. safety, liveness, fairness), and quality properties (e.g. agility, performance). We will develop different complementary analysis techniques combining simulation, model checking, and testing. The aim is to enable analysis and validation of a SoS architecture anytime along the whole SoS life-cycle by means of automated verification. Automated verification will include infinite-state model checking and abstract interpretation techniques for the critical parts of SoS, and testing for non-critical parts to ensure the correct functioning of SoS.

- Architecture-driven construction of SoSs: SoS architecture will drive the initial construction and subsequent evolutions of a SoS. Initial construction will be developed through refinement, where abstract SoS architectures are refined to "meet in the middle" with concrete selection and integration of discovered systems, consequently defining concrete SoS architectures. In our approach, significant progress beyond the state-of-the-art on SoS construction will be achieved by conceiving abstractions and mechanisms for expressing architecture transformations, where the application of these transformations will support refinement from abstract to concrete SoS architectures, taking into account discovered SoSs. Each transformation can be seen as an architecture rewrite in a formal rewriting system. These transformations are enforced to be refinements if preconditions are satisfied and proof obligations discarded. Concrete architectures are deployed to directly implement running SoSs in terms of middleware-based glue code.

- Evolution of SoS architectures and underlying SoSs: an evolving system must be continuously aware of its own behaviour and surrounding environment to execute efficiently and to react to new situations. In the ArchWare Research Project, we will develop SoS-specific concepts and mechanisms of software instrumentation, based on probes and gauges, to implement and support such a continuous feedback. We will utilise a set of probes to observe and quantify significant events as defined by SoS requirements. The

probes will have the ability to be placed into running SoSs. Each SoS will use the input from the probes according to the interpretation of SoS-specific gauges, together with the models of the mission and the architecture to decide when, how, and where evolution is appropriate. For supporting feedback, due to the specific nature of a SoS, monitoring will be required at two distinct levels: the architecture of the SoS and its assigned mission. On the one hand, monitoring the architecture of the SoS implies maintaining a suitable model of all dependencies among the systems involved in the SoS. Those dependencies being of various types, we will formally describe them in such a way that any change in the architecture will be automatically detected. On the other hand, monitoring the mission implies collecting data from the systems comprising the SoS, indicating that the mission is in progress, or detecting any event that must be addressed to decide if one has to re-architect the SoS and/or change the mission itself. Due to the nature of the SoS, a challenge is that the mission monitoring system must be itself evolvable as systems involved in the SoS may evolve over time.

Keywords: Software Architecture, Architecture Description, Architecture Analysis, Software-intensive Systems-of-Systems.

Theme: Architecting Software-intensive Systems-of-Systems

3 New Results

3.1 The Systems-of-Systems Architecture Description Language

Keywords: Architecture Description Language, Systems-of-Systems, Software Architecture.

Participants: Flavio Oquendo, Isabelle Borne, Jérémy Buisson, Régis Fleurquin, Elena Leroux, Salah Sadou, Gersan Moguérou.

The architecture provides the right abstraction level to address the complexity of Software-intensive Systems-of-Systems (SoSs). The research challenges raised by SoSs are fundamentally architectural: they are about how to organize the interactions among the constituent systems to enable the emergence of SoS-wide behaviors and properties derived from local behaviors and properties by acting only on their connections, without being able to act in the constituent systems themselves.

Formal architecture descriptions provide the framework for the design, construction, and dynamic evolution of SoSs.

From the architectural perspective, in single systems, the controlled characteristics of components under the authority of the system architect and the stable notion of connectors linking these components, mostly decided at design-time, is very different from the uncontrolled nature of constituent systems (the SoS architect has no or very limited authority on systems) and the role of connection among systems (in an SoS, connections among constituents are the main architectural elements for enabling emergent behavior to make possible to achieve the mission of an SoS).

The natures of systems architectures (in the sense of architectures of primitive systems) and systems-of-systems are very different.

Systems architectures are described by extension. In the opposite, SoS architectures are described by intension. Systems architectures are described at design-time for developing the system based on design-time components. In the opposite, SoS architectures are defined at run-time for developing the SoS based on discovered constituents. Systems architectures often evolves offline. In the opposite, SoS architectures always evolves online. Systems architectures supports evolution. SoS architectures supports coevolution

We have conceived and defined an Architecture Description Language (ADL) specially designed for specifying the architecture of Software-intensive Systems-of-Systems (SoS). It provides a formal ADL, based on a novel concurrent constraint process calculus, coping with the challenging requirements of SoSs. Architecture descriptions are essential artifacts for (i) modeling systems-of-systems, and (ii) mastering the complexity of SoS by supporting reasoning about properties. In SosADL, the main constructs enable: (i) the specification of constituent systems, (ii) the specification of mediators among constituent systems, (iii) the specification of coalitions of mediated constituent systems.

SoS are constituted by systems. A constituent system of an SoS has its own mission, is operationally independent, is managerially independent, and may independently evolve. A constituent system interacts with its environment via gates. A gate provides an interface between a system and its local environment.

Constituent systems of an SoS are specified by system abstractions via gates, behavior and their assumed/guaranteed properties. Assumptions are assertions about the environment in which the system is placed and that are assumed through the specified gate. Guarantees are assertions derived from the assumptions and the behavior. Behavior satisfies gate assumptions (including protocols) of all gates.

Mediators mediate the constituent systems of an SoS. A mediator has its own purpose and, in the opposite of constituent systems, is operationally dependent of the SoS, is managerially dependent of the SoS, and evolves under control of the SoS.

Mediators among constituent systems of an SoS are specified by mediator abstractions. The SoS has total control on mediators. It creates, evolves or destroys mediators at runtime. Mediators are only known by the SoS. They enable communication, coordination, cooperation, and collaboration.

Coalitions of mediated constituent systems form SoSs. A coalition has its own purpose, may be dynamically formed to fulfill a mission through created emergent behaviors, controls its mediators

System-of-Systems are specified by SoS abstractions. The SoS is abstractly defined in terms of coalition abstractions. SoS are concretized and evolve dynamically at runtime. Laws define the policies for SoS operation and evolution. In SoSs, missions are achieved through the emergent behavior of coalitions.

SosADL, supported by its toolset, has been applied in different case studies and pilot projects for architecting SoS. In particular, it has been applied in a real SoS for architecting a novel Flood Monitoring and Emergence Response SoS. Deployed in the Monjolino River in the City of Sao Carlos, it provides an experimental set for validating the SosADL and its supporting toolset.

3.2 A Novel π-Calculus for Systems-of-Systems

Keywords: π-Calculus, Process Algebra, Systems-of-Systems.

Participants: Flavio Oquendo.

In the case of Architecture Description Languages (ADLs) for describing the architecture of Software-intensive Systems, process calculi have shown to constitute a suitable mathematical foundation for modeling and analyzing system architectures.

ADLs based on process calculi have formalized: (i) components as defined processes in a specific process calculus, e.g. Darwin ADL in FSP, Wright ADL in CSP, and π-ADL in π-Calculus; (ii) connectors as structured bindings defined by explicit channels between communicating processes using a design-time binding mechanism, e.g. shared actions in Darwin/FSP, attachments in Wright/CSP, extruded channel names in π-ADL / π-Calculus.

The communication bindings in ADLs for the description of system architectures is: (i) decided at design-time, (ii) extensionally defined, (iii) unconstrained by local contexts, (iv) unmediated. By the nature of system architectures, the different process calculi capture the needs for the formalization of such ADLs.

However, none of the existing process calculi provides a suitable basis for modeling and analyzing the architecture of SoSs. Needs related to the description of SoS architectures are to represent: (i) systems as local defined processes, (ii) mediation between communicating processes using inferred bindings (i.e. actual bindings are decided at run-time by the SoS). In the case of SoSs, the binding between channels must be: (i) decided at run-time, (ii) intentionally defined, (iii) constrained by local contexts, and (iv) mediated.

We have evaluated the π-calculus (that subsumes other process calculi such as CSP, FSP, and CCS used as the basis of ADL formalization), in its original form and in the enhanced forms that have been developed along the years regarding the architecture description needs for SoS (binding needs to be decided at run-time, constrained by local contexts, intentional, and mediated). The conclusion of our evaluation is that binding in: (i) the basic π-Calculus is endogenous, unconstrained, extensional, and unmediated; (ii) the Fusion Calculus is exogenous, unconstrained, extensional, and unmediated; (iii) the explicit fusion π-F Calculus is exogenous, constrained, extensional, and unmediated; (iv) the concurrent constraints CC-π Calculus is exogenous, constrained, extensional, and unmediated; (v) the Attributed π-calculus is exogenous, constrained, extensional, and unmediated.

We have defined a novel process calculus for providing a suitable formal foundation of an ADL for SoS as none of the existing π-calculi meets the SoS needs.

The design decisions underlying this novel π-Calculus for SoS, named the πSoS-Calculus, are: (i) binding: binding between channels are designed to be exogenous, constrained by local contexts, intentional, and mediated; (ii) expressiveness: it is designed to be computationally complete (Turing completeness); (iii) style: it is designed to be architecture specification-oriented with support for recursion instead of replication; (iv) typing: it is designed to be strongly typed.

The approach for the design of the πSoS-Calculus is to generalize the π-calculus with mediated constraints (mediation is achieved by constraining interactions), encompassing fusions, explicit fusions, concurrent constraints, and subsuming the basic π-calculus. The πSoS-Calculus

is parameterized by a call-by-value data expression language providing expressions to compute values and to impose constraints on interactions.

The SosADL is based on this novel calculus, the πSoS-Calculus.

3.3 The SosADL Type System

Keywords: Architecture Description Language, Systems-of-Systems, Software Architecture, Type System.

Participants: Jérémy Buisson, Gersan Moguérou, Flavio Oquendo.

Among the objectives of SosADL, we aim at designing a set of techniques and tools for the analysis and verification of SoS architectures. As part of this goal, we have initiated the definition of a type system. To start with, we follow the common approach, organizing the type system based on the structure of the language, as defined by the underlying meta-model.

The type system is under development using the Coq proof assistant. To keep this development synchronized with the evolutions of the meta-model, the Coq definitions that define the meta-model are automatically generated from the Ecore definition of SoSmart. To validate the first typing rules, a collection of test cases are type-checked by-hand thanks to Coq.

3.4 Verification support for SosADL

Keywords: Software Architecture, Verification, Model-Checking, Testing, Simulation.

Participants: Elena Leroux, Gersan Moguérou, Flavio Oquendo.

The purpose of this work is to allow formal verification of systems of systems (SoS) described using the SoSADL language by extending the SoSmart SoS Architecture framework with appropriate techniques and tools. Indeed, it is important to guarantee that the important functional properties of SoS are hold during the whole live cycle of this SoS. In order to achieve our purpose, we extract the formal model of a given SoS. We use an Input-Output Symbolic Transition System (IOSTS) representing the architecture structure and behaviors of SoS. Then the idea is to automatically translate the obtained IOSTS of SoS into the XTA format in order to perform simulation, verification and testing of SoS using the UPPAAL tool.

3.5 Architectural Description by Constraints for SosADL

Keywords: Architectural Description, ADL, Constrainst Satisfaction Problem.

Participants: Milena Guessi, Flavio Oquendo, Elisa Nakagawa.

Software systems that have become increasingly large and complex due to the combination of operationally independent and geographically distributed constituent systems are referred to as Systems-of-Systems (SoS). Due to their dynamic, evolving nature, software architectures for SoS must be able to adjust their own structure for coping with changes in their composition (such as addition or removal of constituents) and/or missions.

An adequate description of SoS software architectures is quite important to the success of these systems. In this context, Architecture Description Languages (ADLs) have become well-accepted approaches for systematically representing and analyzing software architectures. One of the most emergent ADLs for describing SoS is SoSADL, which supports the description of constituents, mediators, and their architectural configuration (i.e., the arrangement of constituents and mediators that form an SoS).

Explicitly determining an SoS' architectural configuration could be impractical, or even impossible, due to its size and dynamic nature. In this scenario, a declarative, intentional description of the architectural configuration offers important guidelines for attaching constituents at runtime. Nonetheless, to be effective, an approach for automatically conceiving such architectural configurations is still an open issue.

To address this open issue, we depict the architectural configuration of SoS in SoSADL as a Constraint Satisfaction Problem (CSP) using the Alloy language. In particular, our Alloy meta-model for SoSADL can be used for finding all valid architectural configurations for an SoS within a limited scope. This approach opens interesting perspectives on how specific arrangements among constituents and mediators could impact the quality of SoS. We will complement this approach with transformations from SoSADL to Alloy and vice-versa.

With this approach, we intend to contribute for the discovery and analysis of architectural configurations of SoS. Formalizing the architectural configuration of SoS as a CSP enables one to take advantage of existing tools for automatically finding valid configurations that meet their specifications. In particular, it is also possible to check if a given architectural configuration preserves important properties or, otherwise, present an architectural configuration that violates these properties. As a consequence, our approach should contribute for the correctness of SoS.

3.6 Defining SoS patterns for use with SosADL

Keywords: System-of-Systems, Pattern, Dynamic Reconfiguration.

Participants: Franck Petitdemange, Isabelle Borne, Jérémy Buisson.

This work aims to defining SoS patterns for use with SosADL. It has mainly addressed the state-of-the-art for identifying open issues and proposed approaches. The focus is on patterns for dynamic reconfiguration.

3.7 A meta-process for describing SoS architectures

Keywords: Architectural Construction, System-of-Systems, Architectural Decisions.

Participants: Marcelo Gonçalves Flavio Oquendo, Elisa Nakagawa.

In the System-of-Systems (SoS) context, software architectures have been regarded as an important element for determining the success of such systems, mainly due to their potential of: (i) encompassing the organization of the constituent systems; (ii) addressing the inherent characteristics of SoS; (iii) dealing with functional and non-functional requirements, and

complexity concerns; and (iv) systematizing the principles that drive the evolution of these architectures.

The development of SoS differs from monolithic systems in several issues, such as the dynamic contribution and impact of constituent systems, which are developed and managed by independent sources. In this context, several challenges emerge on architecting SoS, such as the integration of heterogeneous constituent systems and the determination of special communication protocols and integration rules. Therefore, SoS software architectures have reached a threshold in which traditional software engineering approaches are no longer adequate. However, there is a lack with respect to processes and methods to construct SoS considering the design, representation, implementation, and evolution of their architectures.

For addressing these open issues, we proposed a meta-process independent of specific implementation technologies or application domains, named "Meta-process for SoS Software Architectures" (SOAR). With SOAR, software architects, process managers, and other SoS stakeholders can have support to instantiate their own processes when constructing SoS software architectures.

SOAR has been the result from an analysis of the state of the art of SoS development in conjunction with lessons learned with collaborating experts and investigations in real-world development environments. It can be valuable for several application domains and with the maturation of new good practices as standard solutions for SoS, new architectural decisions can be further incorporated to SOAR, yielding new versions for more specific contexts. Moreover, SOAR was produced with OMG's Essence Language through a modularized perspective in which the main scopes of problem concerning the construction of SoS software architectures were dealt (i.e., software development, construction of software architectures, and construction of SoS software architectures). This modularization allows the best understanding and application of SOAR as a meta-process.

In order to evaluate the applicability of SOAR, two teams of SoS software architecture experts were consulted in a survey that considered only the modules relative to the two first scopes of problem. The first one was composed by three experts, which provided improvements to our proposal. Afterwards, a qualitative survey was conducted with a second team formed by six experts external to our research group. These experts have been involved to the development of SoS and software architecture in both academy and industry. The conducted survey aimed to verify if the process requirements were adequate, and if SOAR meets the expectations of the SoS community as a meta-process to support the construction of SoS software architectures. The chosen approach for gathering data was the use of online questionnaires. The collected answers showed us important improvement directions. For example, the specification of different categories of SoS (i.e., virtual, collaborative, acknowledged, and directed). The experts also observed that SOAR and the requirements are clear, their elements are described without ambiguities, and the support material provided with SOAR is sufficient to enable the evaluation team to understand all of the technical terms. In a nutshell, the results pointed out that SOAR meet the process requirements on constructing SoS software architectures, encompassing all general challenges that play a role in the construction of these architectures.

3.8 Architectural design of service-oriented robotic systems and systems-of-systems

Keywords: Service Oriented Architecture, System-of-Systems, Architectural Decisions.

Participants: Lucas Oliveira, Flavio Oquendo, Elisa Nakagawa.

Robots have increasingly supported different areas of the society, making daily tasks easier and executing dangerous, complex activities. Due to this high demand, robotic systems used to control robots are becoming larger and more complex, which creates a great challenge to the development of this special type of software system. Over the last years, researchers have been adopting the Service-Oriented Architecture (SOA) architectural style as a solution to develop more reusable, flexible robotic systems. Several studies in the literature report the creation of Service-Oriented Robotic Systems (SORS), as well as new languages and environments for the development of such systems. Nevertheless, few attention has been paid to the development of SORS software architectures. Currently, most of software architectures are designed in an *ad hoc* manner, without a systematic approach of development, hampering the construction, maintenance, and reuse of robotic systems. The consideration of quality attributes since the software architecture design is a critical concern, as these systems are often used in safety-critical contexts.

To mitigate this issue, we first defined a process named ArchSORS (Architectural Design of Service-Oriented Robotic System), which aims at filling the gap between the systematic development of service-oriented systems and the current *ad hoc* approaches used to develop SORS. The ArchSORS process provides prescriptive guidance from the system specification to architecture evaluation. Following, we established a reference architecture to support the design of SORS software architectures developed using ArchSORS. The reference architecture, named RefSORS (Reference Architecture for Service-Oriented Robotic System), is based on different sources of information, such as: (i) SORS available in the literature identified by means of a systematic review, (ii) a taxonomy of services for developing SORS, established in conjunction of robotics specialists, (iii) reference models and reference architectures available for SOA, (iv) reference architectures for the robotics domain, and (v) control architectures of the robotics domain. RefSORS encompasses multiple architecture views described in high-level representations and the semi-formal description languages SoaML (Service-oriented architecture Modeling Language) and UML (Unified Modeling Language).

Results of a controlled experiment involving students engaged in the French robotics competition RobaAFIS already pointed out that ArchSORS can improve coupling, cohesion, and modularity of SORS software architectures, which can result in systems of higher quality. The same RobaAFIS project adopted in the experiment was also developed in the context of a case study that uses the RefSORS reference architecture to support the application of the ArchSORS process. The software architecture designed in this case study presented better results in the three metrics (coupling, cohesion, and modularity) if compared to those created in the experiment by using only the ArchSORS process. The robotic system designed during the case study is currently under development.

Theme: Architecting Dynamic Software-intensive Systems

3.9 Architecture-based code generation for dynamic software-intensive systems

Keywords: Architecture description languages, π-ADL, Implementation, Go.

Participants: Everton Cavalcante, Flavio Oquendo, Thais Batista.

A recurrent problem of almost all Architecture Description Languages (ADLs) is the decoupling between architecture descriptions and their respective implementations. As software architectures are typically defined independently from implementation, ADLs are often disconnected from the runtime level, thus entailing mismatches and inconsistencies between architecture and implementation mainly as the architecture evolves. Even though a system is initially built to conform to its intended architecture, its implementation may become inconsistent with the original architecture over time. This problem becomes worse with the emergence of new generation programming languages because most ADLs do not capture the new features of this type of languages, which are intended to take advantage of the modern multicore and networked computer architectures. Therefore, the architectural representation and system implementation need to be continuously synchronized in order to avoid architectural erosion and drift.

In order to support the transition from architecture description to implementation, the π-ADL was integrated with the Go language (http://golang.org/), a new programming language recently developed at Google. Unlike most existing ADLs, π-ADL provides a formal language for describing dynamic software architectures by encompassing structural and behavioral viewpoints, as well as supporting their automated, rigorous analysis with respect to functional and non-functional properties. In turn, Go was chosen to serve as implementation language because it is an easy general-purpose language designed to address the construction of scalable distributed systems and to handle multicore and networked computer architectures, as required by new generation software systems. The integration of π-ADL with Go was mainly fostered by their common basis on the π-calculus process algebra and the straightforward relationship between elements of the languages, such as the use of connections in π-ADL and channels in Go as means of communication between concurrent processes.

The integration of π-ADL and Go resulted in comprehensive correspondences between the languages, which were used to develop a technical process aimed to automatically source code in Go from architecture descriptions in π-ADL. Furthermore, an Eclipse-based plug-in was built to assist software architects in the textual description of architectures using the π-ADL language and to generate source code in Go. Therefore, when describing a software architecture in π-ADL, if it is correct according to the syntactic and semantic rules of the language (verified by the textual editor), then the respective Go source code is generated with the automatic build capability provided by the π-ADL textual editor.

3.10 Supporting dynamic reconfiguration of software-intensive systems with Coqcots & Pycots

Keywords: Software Architecture, Component Based Systems, Dynamic Reconfiguration.

Participants: Jérémy Buisson, Elena Leroux, Everton Calvacante, Fabien Dagnat

(Télécom Bretagne), Sébastien Martinez (Télécom Bretagne).

Dynamic reconfiguration of component-based software systems is well-established. Based on quiescence, the main approach (such as FRACTAL, OPENCOM) consists in suspending and resuming some of the components. In practice this approach requires suspending the components that depends on suspended components, so the suspension propagates through the whole architecture. The most common alternative approach (such as OSGi) consists in considering all the dependencies as optional. This approach is impractical too because component implementations are expected check for the dependencies and must provide some behaviour in case the dependencies are not satisfied.

To solve this issue, we propose to use Dynamic Software Updating in order to change the implementation and type of components at runtime. That way, the behaviour of the components can be changed temporarily to accommodate missing dependencies during the time of reconfiguration.

The semantics of the reconfiguration actions has been mechanized using the Coq proof assistant. This work turns the Coq language into a reconfiguration language. On the one side, the proof assistant allows to formally verify the reconfiguration. For instance proof obligations are issued for the preconditions of each reconfiguration action. On the other side, the extraction mechanism of Coq is used to generate the executable reconfiguration script.

To validate our work, we have used Coq to formally prove that our semantics is consistent with our component model. As a first step we have applied the whole process to a toy web server using a Python implementation of the component-model and our Pymoult Dynamic Software Updating platform.

3.11 Categorization of dynamic software updating mechanisms

Keywords: Dynamic Software Updating.

Participants: Sébastien Martinez (Télécom Bretagne), Fabien Dagnat (Télécom Bretagne), Jérémy Buisson.

Implementing dynamic software update (DSU) mechanisms from the literature in Pymoult led us to a design easing their combination and configuration. While state of the art DSU platforms use a selected combination of mechanisms to enable dynamic updates, Pymoult allows developers to select the mechanisms they find best suited for each update. While it gives better control on the updates to the developer, this strategy is less restrictive on application compatibles with DSU. Pymoult was successfully used to update Django powered applications.

We consider the design of Pymoult to be the vanguard of a generic API for DSU that could be followed by next generation DSU platforms. Such DSU platforms could be combined to update complex applications for which a single platform would not be sufficient (*e.g.* multi-language applications).

3.12 Formal verification and generation for reconfigurable socio-technical systems

Keywords: formal verification, requirements modelling, model checking.

Participants: Soraya Mesli, Pascal Berruet (Lab-STICC), Alain Bignon (SEGULA), Flavio Oquendo.

The design of socio-technical systems is an activity more and more complex because it involves several designers from very different technical backgrounds. This variety of profiles causes to comprehension difficulties of specifications, which reflects the high number of errors usually detected during the product testing stage. To minimize theses errors in the earlier stages of designing, a model-driven engineering approach was proposed. This approach permits to each designer to concentrate in his heart craft. The bridge between the designers is realized by a succession of models transformations. Most models were generated automatically. The approach cited above makes consistency between the generated models. But it does not allow formal verification of these models. Indeed, if the source model contains design errors due to misinterpretation of the specifications, automatic generation causes the propagation of these errors to the generated models. The main goal of this work is to introduce in the earlier stages of the aforementioned approach a set of formal verification techniques to obtain secure systems with reduced cost and respecting the customer requirements.

Our proposed approach is divided in three steps. The goal of each step is expressed by a question. These tree questions are: (i) What we should verify?; (ii) How we should verify ?;(iii) Where we should verify?

What we should verify? The goal of this step is to determine the most important properties that should be verified. To get a list of all important properties, we have made four semi structured interview with different designers. These interviews permit us to establish an initial list of properties. This initial list will be completed by the state of art of common verified properties. At the end, we will obtain a complete list of properties that should be verified.

How we should verify? The purpose of this question is to choose among all the existing verifications methods, the more adequate of kind of property and our system. We have elaborated a protocol of systematic mapping in this field.

Where we should verify? The main of this step is to determine what type of property should be formally verified on which system?s model. The complete list will allow us to make this decision. System?s model should be translated into a mathematical model. We will use model checking to verify the property on the model.

The tree steps cited above will allow us to determine What, How and Where to verify. This work is a foundation for the rest of our work. Indeed, the list of properties will be rewritten in mathematical form, the system?s model will also be translated in mathematical model. We will use model checking method to verify that the model satisfies the property or no. The next step of our work is to think and propose the algorithms based on the MDE concepts, to transform system?s model in mathematical model. The goal of our work is to generate the formal verification automatically.

Theme: Supporting the architecture of software-intensive systems

3.13 Preserving Architectural Decisions through Architectural Patterns

Keywords: Architectural decision, Pattern, Pattern composition, Model Driving Engineering..

Participants: Minh Tu Ton That, Salah Sadou, Flavio Oquendo, Isabelle Borne.

Architectural decisions have emerged as a means to maintain the quality of the architecture during its evolution. One of the most important decisions made by architects are those about the design approach such as the use of patterns or styles in the architecture. The structural nature of this type of decisions give them the potential to be controlled systematically. In the literature, there are some works on the automation of architectural decision violation checking. In this project we show that these works do not allow to detect all possible architectural decision violations. To solve this problem we proposed an approach which: i) describes architectural patterns that hold the architectural decision definition, ii) integrates architectural decisions into an architectural model and, iii) automates the architectural decision conformance checking. The approach is implemented using Eclipse Modeling Framework and its accompanying technologies. Starting from well-known architectural patterns, we shown that we can formalize all those related to the structural aspect. Through two experiments, we shown that architectural decisions are well explained and all of their violations are detected.

Composable software systems have been proved to support the adaptation to new requirements thanks to their flexibility. A typical method of composable software development is to select and combine a number of patterns that address the expected quality requirements. A lot of work have shown the interest of pattern composition. Nevertheless, one of the shortcomings of these work is the vaporization of composition information which leads to the problem of traceability and reconstructability of patterns. Our approach also proposes to give first-class status to pattern merging operators to facilitate the preservation of composition information. The approach is tool-supported and an empirical study has also been conducted to highlight its interests.

3.14 Make reusable components extracted from object-oriented applications

Keywords: Component-based Architectures, Legacy Systems, Restructuring, Meta-Heuristic Approach.

Participants: Abderrahmane Seriai, Salah Sadou, Houari Sahraoui (Univ. of Montreal).

Reuse is one of the main goals of software engineering. Many concepts and associated mechanisms were proposed to promote the reuse of features offered by the software. Component based software engineering (CBSE) is one of the important approaches which was proposed to increase the software reuse. Thus, several works were conducted in order to restructure legacy systems into component-based ones. Nevertheless, almost all proposed approaches are more targeted for component identification than for the identification of reusable parts. Thus, they are limited to identify components, corresponding to sets of classes, allowing an extraction of the architectural view of the legacy system. But, the identified components can not be easily implemented in a concrete component model.

In this work, we propose two approaches to improve the reusability of the extracted components and by the way facilitate the comprehension of the underlying architecture. Thus, the first approach aims the identification of the extracted components interfaces according to its

interactions with the other components. The second one aims to make extracted components implementable within a concrete component model. This is done by using class instances (objects) that compose the extracted components to infer possible component instances.

The evaluation of the proposed approaches via an empirical study showed that i) overall the identified interfaces correspond to the different functional aspects of the extracted components. ii) and, that is possible to implement, within a concrete component model, the extracted components without altering the behavior of the application.

3.15 Using Object-oriented metrics on component-based applications

Keywords: Software Architecture, Component Properties, Object-oriented Metrics, OSGI Framework, Software Quality.

Participants: Salma Hamza, Salah Sadou, Régis Fleurquin.

Over the past decade, many companies have introduced component-oriented software technology in their software development. In this paradigm, as in all others, architects and developers need to evaluate as soon as possible the quality of what they produce, especially during the process of designing and coding. Code metrics are useful tools to do this. These internal metrics can help to predict the expected external quality of components or architectures being encoded. Several proposals for metrics have been made in the literature for the component world. Unfortunately, none of the proposed metrics have been seriously study regarding: their completeness, soundness and their ability to effectively predict the external quality of the developed artifacts. Even worse, the lack of their support by the available code analysis tools makes it impossible their industrial use. Then, the quantitatively prediction of the external quality is impossible. The risk is therefore high to increase the costs consequence of the late discovery of defects.

Based on the premise that much of the industrial component frameworks are based on object-oriented technology, we have studied the possibility of using some imperative and object-oriented code metrics, to evaluate component-based applications. Indeed, these metrics have the advantage of being well defined and known and supported by many tools. Among these metrics, we have identified a subset which can be interpreted and applied to the 3 specific point of view of component (internal, interface and composition). This suite of metrics was applied to 10 open-source OSGi applications, in order to ensure, by studying of their distribution, that they conveyed effectively relevant information to the component world. This study also provides an opportunity to discuss the importance and the respect of some software engineering principles by developers and architects of OSGi worldwide. Based on these metrics, we also have built several predictive models for two external quality properties: number of revisions and failure. To do this, we relied on external data from the history of changes of 6 large and mature OSGi projects (with a maintenance period of several years). Several statistical tools were used to build these models, including principal component analysis and multivariate logistic regression. This study showed that it is possible to predict with these models 80% to 92% of frequently buggy components with reminders ranging from 89% to 98%, according to the evaluated project. Our study has shown that some of the imperative and object-oriented metrics can be used by developers and architects to check the quality of component-based

applications.

3.16 Development of an architectural framework for business intelligence

Keywords: Service-oriented architecture, Business intelligence, Budget planning.

Participants: Davy Hélard, Jean-Philippe Gouigoux (MGDIS), Flavio Oquendo.

Our aim is to develop of an architectural framework for enabling the developemnt of model-based applications in business intelligence (BI). In particular, we have addressed the issue of architecting model-based systems taking into account different viewpoints. For instance, in BI, budget planning is carried from many different viewpoints, e.g. the investment budget and the operating budget. The first one could be based on a monthly described model and the other one on a weekly described model. Modeling with a spreadsheet force the model to be express for one periodicity and do not allow to evaluate the model for other periodicity. So this makes hard to automatically keep the two parts consistent with each other. Besides, this consistence is import in order to analyze the overall. Of course, it can be done manually but it is time waste and error prone.

To get rid of the constraints that come with spreadsheet, a medialization that is inspired by the mathematical continuous functions is proposed. The nature of continuous function allows to define the model without take into account periodicity issues. The periodicity is only chosen at the evaluation.

To come back to the example, the investment part can be evaluated by month for a user interface (from one viewoint) and also evaluated by week for the evaluation of the operating part (from another viewoint).

The consistency is enforced by a novel service-oriented architecture that segregates the model and the different evaluations, supporting concurrent excetions from different viewpoints.

3.17 Development of an architectural framework for business intelligence

Keywords: Service-oriented architecture, Business intelligence, Budget planning.

Participants: Tahar Gherbi , Isabelle Borne.

Mobile agents facilitate the implementation of dynamically adaptable applications and provide a generic framework to develop distributed applications on large networks. Generally, the development of mobile-agents applications is done without considering the "mobility" aspect in the analysis and design phases; this aspect is often treated in the implementation phase. Considering it earlier (i.e., in the analysis and design phases), improves the quality and reliability of applications. However, little research has focused on methods and tools of analysis and design of mobiles-agents applications. According to literature, modeling these applications can be done with three approaches: design patterns approach, formal approach and semi formal approach which includes formalisms extending UML notations and approaches extending multi-agents systems development methodologies. Because multi-agents systems are relevant to many applications, we are interested in extending multi-agents systems development methodologies to support mobility. Besides, model driven engineering helps to reduce

the gap between multi-agents systems development methodologies (as the majority does not include an implementation phase) and runtime platforms. It also facilitates the moves of mobiles-agents across heterogeneous platforms by transferring the agentÕs model rather than its code. Consequently, this work considers "mobility" in the design phase and proposes a model driven engineering approach to develop multi-agents systems supporting mobility.

A MDE approach has been proposed to develop multi-agent systems (MAS) including mobile agents. The objective is to consider the "mobility" aspect in the design phase rather than considering it only in the implementation phase, as usually done, in order to provide the designer with the ability to model mobility and then to fulfill the objectives of mobile-agents applications. This work combines three technologies (MAS, mobile agents and MDE) to profit from all their benefits; particularly, exploiting the relevance of MAS to design distributed, complex and robust applications, enhancing the adaptation ability of applications, automating the development process, breaking free of heterogeneity constraints, reducing the gap between the MAS development methodologies (which the majority does not contain the implementation phase) and the agentsÕ platforms and easing the agentÕs moves between heterogeneous platforms by sending the agentÕs model rather than its code. Despite these benefits, we have encountered no approach based on MDE and extending a MAS development methodology to support mobility. This work proposed such an approach. As a Platform-Independent Model (PIM) meta-model, we have proposed a meta-model which is simple, modular, general, scalable and which support mobility. Its use was illustrated by modeling a simple application example (book locations search through a network). This later has also been modeled using the meta-models proposed in m-MaSE, m-Gaia and MAGR to show their limits and situate our meta-model with them. JavAct, a mobile-agents platform, has served as runtime platform for our experience; therefore, we have elaborated its Platform Specific Model (PSM) meta-model as well as transformation rules from a PIM (conform to our PIM meta-model) to a PSM (conform to JavActÕs PSM meta-model). The tool Papyrus/Eclipse has served to elaborate the PIM, PSM and theirs meta-models, the tool ATL/Eclipse has served to write and execute the transformation rules and the tool Acceleo/Eclipse has served to write and execute the code generation rules from the obtained PSM to JavAct. PIM-to-PSM transformation and PSM-to-JavAct code generation were illustrated using the same application example (book locations search through a network).

4 Software

4.1 SosADL IDE: The SoSmart Architect Studio

Participants: Gersan Moguérou, Jérémy Buisson, Elena Leroux, Flavio Oquendo.

SoSmart is a novel environment for description, analysis, simulation, and compilation/execution of SoS architectures. These SoS architectures are described using the SosADL language, which is a language based on process algebra, and on a meta-model defining SoS concepts. Because constituents of an SoS are not known at construction time, the language promotes a declarative approach of architecture families. At runtime, the SoS evolves within such a family depending on the discovery of concrete constituents.

At the end of 2014, the language has being heavily discussed, and improved (concrete syntax, metamodel, type system). The development of an IDE has been started using Eclipse Xtext. The sources are now hosted on INRIA's forge, and available to the whole team.

The formal verification tools are under development, using the intermediate IoSTS language, which has been extended in order to be a convenient pivot language between SoSADL and other model-checking languages like XTA from the Uppaal tool.

Two Eclipse plugins have been developed and integrated to the SosADL IDE:

- Mm-to-Coq that makes a bridge from an Ecore meta-model to Coq.
- org.archware.sosadl.coq that converts an SoSADL file to Coq, using the artifacts generated by Mm-to-Coq.

The PhD students are now able to use the SosADL IDE for their own studies: dynamic architectures, architectural styles, automatic generation of architectures.

4.2 SoSALD2IOSTS

Participants: Elena Leroux, Gersan Moguérou.

SoSADL2IOSTS is a part of the SoSmart SoS Architecture framework which purpose is to represent the behaviors of systems of systems expressed using the SoSADL language as IOSTS models in order to verify interesting and important functional properties of SoS by giving this model to different existing tools used for verification of software systems.

4.3 Coqcots & Pycots

Participants: Jérémy Buisson, Elena Leroux, Fabien Dagnat (Télécom Bretagne), Sébastien Martinez (Télécom Bretagne).

Coqcots & Pycots is a component model and framework under development on the gforge Inria as a collaborative work with the PASS team. Coqcots is a Coq model that allows the reconfiguration developer to formally specify, program then verify reconfiguration scripts. Pycots is the corresponding framework for the Python language.

The process is supported end-to-end. The architecture of a running Pycots application can be extracted using a reflexive level. This extraction builds a Coqcots model, a formal component model defined in Coq. Once the architecture is extracted, the developer simultaneously defines its reconfiguration and proves its correctness within Coq. Once proved, the reconfiguration script is extracted from the proof using Coq extraction facilities. Then this script is glued with DSU code developed to support the updates of component behavior. Lastly, this code is uploaded to the Pycots running application to be executed. The resulting update therefore modify the application without stopping it.

Pycots relies on Pymoult, a library developed at Télécom Bretagne, which provides many DSU mechanisms in a single platform.

As part of the project, the extraction plugin of Coq has been extended in order to support Python output.

4.4 Pymoult

Participants: Sébastien Martinez (Télécom Bretagne), Fabien Dagnat (Télécom Bretagne), Jérémy Buisson.

Pymoult is a Pypy library providing a prototyping platform for Dynamic Software Updating. Many of the mechanisms from the literature has been reimplemented in Pymoult. The library then allows to recombine these mechanisms at developers' wish in order either to simulate other existing platforms or to experiment new combinations. Currently, more than 15 existing platforms can be simulated with Pymoult.

4.5 ADManager

Participants: Minh Tu Ton That, Salah Sadou, Flavio Oquendo..

ADManager is a tool to document, by using pattern model, solutions in the architecture that are related to some architectural decisions (AD). In ADManager tool, one can define a pattern model, create an AD by mapping the pattern model to the architectural model and verify the conformance of the architectural model against the created solutions to AD after a change in the architecture.

4.6 COMLAN

Participants: Minh Tu Ton That, Salah Sadou, Flavio Oquendo..

COMLAN is a tool to design architectural patterns with the focus on documenting pattern composition operations. In COMLAN tool, merging operators are used as model element. Specifically, one can store merging operators in a persistent form, reference to merging operator from another element, etc.

Thus, COMLAN provides the following functionalities:

- Create architectural patterns
- Compose patterns using merging operators
- Refine the composed pattern

5 Contracts and Grants with Industry

5.1 Grants Involving Industry

Collaboration on systems for processing big-data (CIFRE)
Participants: MGDIS.

Collaboration on analysis of systems-of-systems (CIFRE)
Participants: SEGULA.

5.2 International Grants - Cooperative Projects

SASoS (Supporting Development of Software Architectures for Software-intensive Systems-of-Systems)

- Funding: FAPESP (Sao Paulo State Research Agency)
- Period: 2014 - 2016
- Partners: University of Sao Paulo - ICMC Research Institue (Brazil)
- Objective: To develop a framework for architecting Software-intensive Systems-of-Systems.

ArchIoT (Software Architecture for the Internet-of-Things)

- Funding: INES-CNPq (National Institute for Software Engineering - National Research Agency)
- Period: 2014 - 2015
- Partners: UFRN - Federal University of Rio Grande do Norte (Brazil)
- Objective: To develop an ADL based on SysML for architecting applications to be deployed on the Internet-of-Things.

ProSA-RAES4D (Framework to Describe Reference Architectures for Embedded Systems)

- Funding: FAPESP (Sao Paulo State Research Agency)
- Period: 2012 - 2014
- Partners: Fraunhofer-Institut für Experimentelles Software Engineering IESE (Germany), University of Sao Paulo - ICMC Research Institute (Brazil)
- Objective: To develop a formal language and framework for describing reference architectures of embedded systems.

ProSA-RAES4A (Framework to Analyze Reference Architectures for Embedded Systems)

- Funding: CNPq (National Research Agency)
- Period: 2012 - 2014
- Partners: University of Sao Paulo - ICMC Research Institute (Brazil)
- Objective: To develop a formal language and framework for analyzing reference architectures of embedded systems.

6 Other Grants and Activities

6.1 International Collaborations

- Flavio Oquendo:
  - University of Sao Paulo - ICMC Research Institute, Sao Carlos, Brazil (Elisa Nakagawa): Architecting critical systems-of-embedded systems (PhDs in co-tutelle)
  - UFRN - Federal University of Rio Grande do Norte, Natal, Brazil (Thais Batista): Architecting dynamic software-intensive systems-of-systems (PhD in co-tutelle)
- Salah Sadou:
  - University of Montréal (Houari Sahraoui): Restructuring object-oriented systems into component-based systems (PhD in co-tutelle)
  - University of Science and Technology of Houari Boumedienne, Alger, Algeria (Mohamed Ahmed Nacer): Product line architecture for systems-of-systems (PhD in co-tutelle)
- Isabelle Borne:
  - LISCO, University Badji Mokhtar Annaba, Algeria (Djamel Meslati): A model driven engineering approach to design mobile agent applications (PhD in co-tutelle)

6.2 National Collaborations

- Jérémy Buisson is an external collaborator of Télécom Bretagne (PASS), contributing to the supervision of the PhD student Sébastien Martinez
- Flavio Oquendo has a collaboration on systems-of-systems with Khalil Drira (LAAS-CNRS) and Axel Legay (INRIA)
- Salah Sadou has a collaboration on reuse of architectural constraints with Chouki Tibermancine and Christophe Dony (LIRMM)

7 Dissemination

7.1 Editorial Boards and Guest Editions

- Flavio Oquendo:
  - Springer Journal of Software Engineering Research and Development (Member of the Editorial Board)
  - International Journal of Artificial Intelligence and Applications for Smart Devices (Member of the Editorial Board)
  - International Journal on Advances in Software (Member of the Editorial Board)

  - Special Issue on Software Components, Architectures and Reuse of the International Journal of Universal Computer Science (Guest Editor)
  - Special Issue on Adaptive and Reconfigurable Service-oriented and Component-based Applications and Architectures of the Inderscience International Journal of Autonomous and Adaptive Communications Systems (Guest Editor)
  - Special Issue on Advanced Architectures for the Future Generation of Software-intensive Systems of the Elsevier Journal on Future Generation Computer Systems (Guest Editor)

- Régis Fleurquin:

  - Special Issue on Advanced Architectures for the Future Generation of Software-intensive Systems of the Elsevier Journal on Future Generation Computer Systems - FGCS (Referee of the Special Issue)

7.2 General Chairs, Steering Committees

- Flavio Oquendo:

  - European Conference on Software Architecture - ECSA (Steering Committee Chair)
  - IEEE/IFIP Working International Conference on Software Architecture - WICSA (Steering Committee Member)
  - Conférence francophone sur les architectures logicielles - CAL (Steering Committee Member)
  - IEEE International Conference on Collaboration Technologies and Infrastructures - WETICE (Steering Committee Member)
  - ACM International Workshop on Software Engineering for Systems-of-Systems (technically co-sponsored by ACM SIGSOFT and ACM SIGPLAN) - SESOS (Steering Committee Chair)
  - Workshop on Distributed Development of Software, Ecosystems and Systems-of-Systems - WDES (Steering Committee Member)

7.3 Program Chairs, Tutorial Chairs, Special Session Chairs

- Flavio Oquendo: International Workshop on Software Engineering for Systems-of-Systems - SESOS 2014 (Program Co-chair)

7.4 Program Committees

- Isabelle Borne:

  - SESOS: International Workshop on Software Engineering for Systems-of-Systems, 2014

  - WETICE: IIEEE International Conference on Collaboration Technologies and Infrastructures, 2014
  - CIEL: 3eme Conference en IngenieriE du Logiciel, 2014

- Jérémy Buisson:
  - ICCS: International Conference on Computational Science, 2014
  - C&ESAR: Computer & Electronics Security Applications Rendez-vous, 2014

- Régis Fleurquin
  - TSI Journal "L'objet": served as reviewer in 2014

- Flavio Oquendo:
  - WICSA: IEEE/IFIP Working International Conference on Software Architecture, 2014
  - ECSA: European Conference on Software Architecture, 2014
  - WCCS: World Conference on Complex Systems, 2014
  - WETICE: IEEE International Conference on Collaboration Technologies and Infrastructures, 2014
  - ICSEA: International Conference on Software Engineering Advances, 2014
  - ICSOFT-EA: International Conference on Software Engineering and Applications, 2014
  - I-ESA: International Conference on Interoperability for Enterprise Systems and Applications, 2014
  - AROSA: IEEE WETICE Conference Track on Adaptive and Reconfigurable Service-oriented and component-based Applications and Architectures, 2014
  - PESOS: International Workshop on Principles of Engineering Service-Oriented Systems at ACM/IEEE International Conference on Software Engineering (ICSE), 2014
  - IWSECO-WEA 2014: Joint International Workshop on Software Ecosystems & International Workshop on Ecosystem Architectures, 2014
  - DANCE: Workshop on Distributed Architecture Modeling for Novel Component based Embedded Systems at ACM/IFIP/USENIX Middleware, 2014
  - ICAART: International Conference on Agents and Artificial Intelligence, 2014
  - ICSOFT-PT: International Conference on Software Paradigm Trends, 2014
  - ADAPTIVE: International Conference on Adaptive and Self-Adaptive Systems and Applications, 2014
  - ADVCOMP: International Conference on Advanced Engineering Computing and Applications in Sciences, 2014
  - CAL: Conférence Francophone sur les Architectures Logicielles, 2014

  - SBES: National Symposium on Software Engineering, 2014
  - SBCARS: National Symposium on Software Components, Architectures and Reuse, 2014
  - WDES: Workshop on Distributed Development of Software, Ecosystems and Systems-of-Systems, 2014

- Salah Sadou:

  - CBSE: International ACM SigSoft Symposium on Component Based Software Engineering, 2014
  - WETICE: IEEE International Conference on Collaboration Technologies and Infrastructures, 2014
  - SESOS: International Workshop on Software Engineering for Systems-of-Systems, 2014
  - QUORS: IEEE International Workshop on Quality Oriented Reuse of Software, 2014
  - CAL: Conférence Francophone sur les Architectures Logicielles, 2014
  - Journal of Systems and Software (Elsevier): served as reviewer in 2014

7.5 Doctoral Examination Boards

- Flavio Oquendo:

  - Doctoral Examination Board of Minh Tu Ton That (Preserving Architectural Decisions through Architectural Patterns), IRISA, UBS, 2014

- Salah Sadou:

  - Doctoral Examination Board of Minh Tu Ton That, IRISA, UBS, 2014
  - Doctoral Examination Board (as Rapporteur) of André Cavalcante Hora, Univ. Lille 1, 2014
  - Doctoral Examination Board (as Rapporteur) of Rafat Al-Msie'deen, Univ. Montpellier 3, 2014

- Isabelle Borne:

  - Doctoral Examination Board (rapporteur) of Mayleen Lacouture (A chemical programming language for orchestrating Services), Nantes University and Ecole des Mines de Nantes, October 31st 2014.
  - Doctoral Examination Board (President) of Riad Belkhatir (Contribution à l'automatisation et à l'évaluation des architectures logicielles ouvertes), Université de Nantes, June 18 2014.

- Doctoral Examination Board (superviser) of Tahar Gherbi (Une démarche d'ingénierie dirigée par les modèles pour le développement des applications d'agents mobiles), Bretagne-Sud University, June 24 2014.
- Doctoral Examination Board of Salma Hamza, Bretagne-Sud Universty, December 19 2014

7.6 Scientific Networks

- Isabelle Borne: Co-chair of Action IDM (Model-Driven Engineering) - GDR GPL & ASR
- Flavio Oquendo: Co-chair of GT SdS (Systems-of-Systems)- GDR GPL (being created)
- Salah Sadou: Co-chair of GT RIMEL (Reverse-engineering, Maintenance and Evolution of Software) - GDR GPL

7.7 Industrial Collaborations

- Flavio Oquendo: Collaboration with MGDIS
- Flavio Oquendo: Collaboration with SEGULA

7.8 Visiting Positions

- Flavio Oquendo: University of Rio Grande do Norte, Brazil

7.9 Research Excellence Awards (PES)

- Flavio Oquendo: PES A (2011-2015)
- Salah Sadou: PEDR A (2014-2018)

7.10 Laboratory Responsabilities

- Isabelle Borne: Responsible of the Site of Vannes for IRISA

7.11 Teaching Responsabilities

- Régis Fleurquin: Head of Computing Department of IUT (UBS)
- Salah Sadou: Head of Computing Degree of ENSIBS School of Engineering (UBS, from September 2014)
- Flavio Oquendo: Head of Computing Degree of ENSIBS School of Engineering (UBS, until August 2014)
- Flavio Oquendo: Member of the Steering Board of ENSIBS School of Engineering (UBS)

7.12 Scientific Committees

- Flavio Oquendo: Member of the Commitee of Experts of the IDEX Université de Bretagne Loire
- Salah Sadou: Member of the Selection and Validation Committee (CSV) of Pôle Images et Réseaux

7.13 Expertises

- Flavio Oquendo: Expert acting as reviewer and evaluator of R&D Projects for the European Commission (Framework Program 7 & Horizon H2020) for:
  - Software-intensive Systems Engineering,
  - Systems-of-Systems, and
  - Cybersecurity & Trustworthy ICT.

7.14 Academic Council (CA), National Council of Universities (CNU)

- Isabelle Borne: Member of CNU (Conseil national des universités) 27
- Salah Sadou: Member of the CA (Commission recherche du conseil académique) of UBS

8 Bibliography

Books and Monographs

[1] K. Drira, F. Oquendo, *Special Issue on Advanced Architectures for the Future Generation of Software-Intensive Systems*, *International Journal on Future Generation Computer Systems*, 2014, `https://hal.archives-ouvertes.fr/hal-01113563`.

[2] M. Fantinato, U. Kulesza, F. Oquendo, *Special Issue on Software Components, Architectures and Reuse: Software Product Line Engineering and Source Code Enhancements*, *International Journal of Universal Computer Science*, 2014, `https://hal.archives-ouvertes.fr/hal-01114148`.

[3] F. Oquendo, P. Avgeriou, E. Cuesta, Carlos, K. DRIRA, J. C. Maldonado, Y. Nakagawa, Elisa, A. Zisman, *Software Engineering for Systems-of-Systems: Proceedings of the ACM Sigsoft/Sigplan International Workshop SESoS'2014*, *Proceedings of the 2014 European Conference on Software Architecture Workshops, Vienna, Austria*, 2014, `https://hal.archives-ouvertes.fr/hal-01114140`.

[4] B. Rodriguez, Ismael, S. Kallel, F. Oquendo, *Special Issue on Adaptive and Reconfigurable Service-oriented and Component-based Applications and Architectures*, *International Journal of Autonomous and Adaptive Communications Systems (Inderscience)*, 2014, `https://hal.archives-ouvertes.fr/hal-01114149`.

Articles in referred journals and book chapters

[5] R. S. Huergo, P. F. Pires, F. C. Delicato, B. Costa, E. Cavalcante, T. Batista, "A Systematic Survey of Service Identification Methods", *Service Oriented Computing and Applications 8*, 3, July 2014, p. 199–219, https://hal.archives-ouvertes.fr/hal-01113408.

[6] P. Maia, T. Batista, E. Cavalcante, A. Baffa, F. C. Delicato, P. F. Pires, A. Zomaya, "A Web Platform for Interconnecting Body Sensors and Improving Health Care", *Procedia Computer Science 40*, December 2014, p. 135–142, https://hal.archives-ouvertes.fr/hal-01113410.

[7] E. Nakagawa, F. Oquendo, J. C. Maldonado, "Architectures de référence : concepts et processus", *in : Architectures logicielles : principes, techniques et outils*, Hermès Sciences, February 2014, p. 1–34, https://hal.archives-ouvertes.fr/hal-00913503.

[8] M. T. Thon That, S. Sadou, F. Oquendo, I. Borne, "Preserving Architectural Pattern Composition Information through Explicit Merging Operators", *Future Generation Computer Systems*, September 2014, p. 1–32, https://hal.archives-ouvertes.fr/hal-01102209.

[9] M. T. Thon That, S. Sadou, F. Oquendo, R. Fleurquin, "Preserving Architectural Decisions through Architectural Patterns", *Journal of Automated Software Engineering*, October 2014, p. 1–41, https://hal.archives-ouvertes.fr/hal-01102187.

[10] E. Yumi Nakagawa, F. Oquendo, J. C. Maldonado, "Reference Architectures", *in : Software Architecture: Principles, Techniques, and Tool*, John Wiley & Sons, 2014, p. 101–122, https://hal.archives-ouvertes.fr/hal-00913505.

[11] T. Zernadji, C. Tibermacine, R. Fleurquin, S. Sadou, "Assistance à l'évolution du logiciel dirigée par la qualité", *in : Évolution et maintenance des systèmes logiciels*, *Chapitre 9*, April 2014, http://hal-lirmm.ccsd.cnrs.fr/lirmm-01104196.

Publications in Conferences and Workshops

[12] A. Almeida, E. Cavalcante, T. Batista, N. Cacho, F. Lopes, "A Component-Based Adaptation Approach for Multi-Cloud Applications", *in : International Workshop on Cross-Cloud Systems (CrossCloud'14)*, IEEE (editor), *Proceedings of the 2014 IEEE Conference on Computer Communications Workshops*, p. 49–54, Toronto, Canada, April 2014, https://hal.archives-ouvertes.fr/hal-01113087.

[13] A. Almeida, F. Dantas, E. Cavalcante, T. Batista, "A Branch-and-Bound Algorithm for Autonomic Adaptation of Multi-Cloud Applications", *in : 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2014)*, IEEE (editor), p. 315–323, Chicago, United States, May 2014, https://hal.archives-ouvertes.fr/hal-01113090.

[14] L. Bueno Ruas De Oliveira, E. Leroux, K. Romero Felizardo, F. Oquendo, E. Yumi Nakagawa, "Towards a Process to Design Architectures of Service-Oriented Robotic Systems", *in : ECSA*, *8627*, p. 218–225, Vienna, Austria, 2014, https://hal.archives-ouvertes.fr/hal-01067337.

[15] L. Bueno Ruas De Oliveira, F. S. Osorio, F. Oquendo, E. Yumi Nakagawa, "Towards a Taxonomy of Services for Developing Service-Oriented Robotic Systems", *in : 26th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, p. 344–349, Vancouver, Canada, July 2014, https://hal.archives-ouvertes.fr/hal-01113224.

[16] J. Buisson, E. Calvacante, F. Dagnat, E. Leroux, S. Martinez, "Coqcots & Pycots: non-stopping components for safe dynamic reconfiguration", *in : CBSE 2014 : proceedings of the 17th international ACM Sigsoft symposium on Component-based software engineering*, p. 1, Lille, France, June 2014, https://hal.archives-ouvertes.fr/hal-00984365.

[17] E. Calvacante, F. Oquendo, T. Batista, "Architecture-Based Code Generation: From π-ADL Architecture Descriptions to Implementations in the Go Language", *in : 8th European Conference on Software Architecture (ECSA 2014)*, S. I. Publishing (editor), *Lecture Notes in Computer Science, 8627*, p. 130–145, Vienna, Austria, August 2014, https://hal.archives-ouvertes.fr/hal-01112357.

[18] M. Gonçalves, E. Calvacante, T. Batista, F. Oquendo, E. Nakagawa, "Towards a Conceptual Model for Software-Intensive System-of-Systems", *in : 2014 IEEE International Conference on Systems, Man and Cybernetics (SMC 2014)*, IEEE (editor), p. 1605–1610, San Diego, United States, October 2014, https://hal.archives-ouvertes.fr/hal-01113173.

[19] M. Guessi, F. Oquendo, E. Yumi Nakagawa, "An Approach for Capturing and Documenting Architectural Decisions of Reference Architectures", *in : 26th International Conference on Software Engineering and Knowledge Engineering (SEKE)*, p. 162–167, Vancouver, Canada, July 2014, https://hal.archives-ouvertes.fr/hal-01113198.

[20] M. Guessi, F. Oquendo, E. Yumi Nakagawa, "Variability Viewpoint to Describe Reference Architectures", *in : 3rd International Workshop on Variability in Software Architecture (VARSA co-located with WICSA)*, p. 6, Sydney, Australia, April 2014, https://hal.archives-ouvertes.fr/hal-01113192.

[21] P. Maia, E. Cavalcante, P. Gomes, T. Batista, F. C. Delicato, P. F. Pires, "On the Development of Systems-of-Systems based on the Internet of Things: A Systematic Mapping", *in : Proceedings of the 2014 European Conference on Software Architecture Workshops*, ACM (editor), Vienna, Austria, August 2014, https://hal.archives-ouvertes.fr/hal-01113406.

[22] V. Neto, M. Guessi, L. Bueno Ruas De Oliveira, F. Oquendo, E. Yumi Nakagawa, "Investigating the Model-Driven Development for Systems-of-Systems", *in : Proceedings of the 2014 European Conference on Software Architecture Workshops (ECSAW)*, p. 8, Vienna, Austria, August 2014, https://hal.archives-ouvertes.fr/hal-01113186.

[23] B. Oliveira, Lucas, B. Martins, Diogo, A. Amaral, Felipe, F. Oquendo, Y. Nakagawa, Elisa, "Automating the Discovery of Services for Service-Oriented Robotic Systems", *in : 11th Latin-American Robotics Symposium (LARS'2014)*, SÃ£o Carlos, Brazil, October 2014, https://hal.archives-ouvertes.fr/hal-01114162.

[24] P. Pires, E. Cavalcante, T. Barros, F. C. Delicato, T. Batista, B. Costa, "A Platform for Integrating Physical Devices in the Internet of Things", *in : Proceedings of the 12th IEEE International Conference on Embedded and Ubiquitous Computing (EUC 2014)*, IEEE (editor), Milan, Italy, August 2014, https://hal.archives-ouvertes.fr/hal-01113382.

[25] D. S. S. Santos, Daniel, B. Oliveira, M. Guessi, F. Oquendo, M. Delamaro, E. Yumi Nakagawa, "Towards the Evaluation of System-of-Systems Software Architectures", *in : 6th Workshop on Distributed Development of Software, Ecosystems and Systems-of-Systems (WDES co-located with CBSoft)*, Maceió, Brazil, September 2014, https://hal.archives-ouvertes.fr/hal-01113239.

[26] A. Seriai, S. Sadou, H. Sahraoui, S. Hamza, "Deriving Component Interfaces after a Restructuring of a Legacy System", *in : Working IEEE/IFIP Conference on Software Architecture (WICSA)*, p. 31 – 40, Sydney, Australia, April 2014, https://hal.archives-ouvertes.fr/hal-01102164.

[27] A. Seriai, S. Sadou, H. Sahraoui, "Enactment of Components Extracted from an Object-Oriented Application", *in : European Conference on Software Architecture (ECSA)*, p. 234 – 249, Viena, Austria, August 2014, https://hal.archives-ouvertes.fr/hal-01102158.

[28] E. Silva, E. Calvacante, T. Batista, F. Oquendo, F. Delicato, P. Pires, "On the Characterization of Missions of Systems-of-Systems", *in : Proceedings of the 2014 European Conference on Software Architecture Workshops*, ACM (editor), Vienna, Austria, August 2014, https://hal.archives-ouvertes.fr/hal-01113243.

[29] E. Yumi Nakagawa, R. Capilla, F. J. Díaz, F. Oquendo, "Towards the Dynamic Evolution of Context-based Systems-of-Systems", *in : 6th Workshop on Distributed Development of Software, Ecosystems and Systems-of-Systems (WDES co-located with CBSoft)*, p. 45–52, Maceió, Brazil, September 2014, https://hal.archives-ouvertes.fr/hal-01113275.

[30] E. Yumi Nakagawa, M. Guessi, J. Carlos Maldonado, D. Feitosa, F. Oquendo, "Consolidating a Process for the Design, Representation, and Evaluation of Reference Architectures", *in : Working IEEE/IFIP Conference on Software Architecture (WICSA)*, p. 143–152, Sydney, Canada, April 2014, https://hal.archives-ouvertes.fr/hal-01113209.

Miscellaneous

[31] K. Drira, F. Oquendo, "Guest Editorial of the Special Issue on Advanced Architectures for the Future Generation of Software-Intensive Systems at the International Journal on Future Generation Computer Systems", 2014, International Journal on Future Generation Computer Systems, https://hal.archives-ouvertes.fr/hal-01114152.

[32] M. Fantinato, U. Kulesza, F. Oquendo, "Guest Editorial of the Special Issue on Software Components, Architectures and Reuse: Software Product Line Engineering and Source Code Enhancements of the International Journal of Universal Computer Science", 2014, International Journal of Universal Computer Science, https://hal.archives-ouvertes.fr/hal-01114154.

[33] B. Rodriguez, Ismael, F. Oquendo, S. Kallel, "Guest Editorial of the Special Issue on Adaptive and Reconfigurable Service-oriented and Component-based Applications and Architectures of the International Journal of Autonomous and Adaptive Communications Systems (Inderscience)", 2014, International Journal of Autonomous and Adaptive Communications Systems (Inderscience), https://hal.archives-ouvertes.fr/hal-01114153.