Activity Report 2017

Research-Team ArchWare

Software Architecture

Architecting Software-intensive Systems and Systems-of-Systems

D4 – Language and Software Engineering

Contents

1 Team composition

Researchers and faculty

Flavio Oquendo, Full Professor, PEDR, Université Bretagne Sud (Head)
Isabelle Borne, Full Professor, Université Bretagne Sud
Nicolas Belloir, Assistant Professor, Ecoles de St-Cyr Coëtquidan
Jérémy Buisson, Assistant Professor, Ecoles de St-Cyr Coëtquidan
Régis Fleurquin, Associate Professor, HDR, Université Bretagne Sud
Elena Leroux, Assistant Professor, Université Bretagne Sud
Salah Sadou, Full Professor, Université Bretagne Sud

Research engineers

Gersan Moguérou, Research Engineer, Université Bretagne Sud

PhD students

Delphine Beaulaton
Raounak Benabidallah
Rymel Benabidallah
Youcef Bouziane
Elyes Cherfa
Imane Cherfa
Lina Garcès
Milena Guessi-Margarido
Soraya Mesli-Kesraoui
Nan Zhang Messe
Valdemar Neto
Paul Perrotin
Franck Petitdemange
Eduardo Silva

Post-Doc

Armel Esnault

Administrative assistant

Sylviane Boisadan, BIATSS, Université Bretagne Sud

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 in particular 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 and 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, especially 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 endeavor 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.

Besides, in SoSs, architectures are designed to fulfill specified missions. Indeed, an important concern in the design of SoSs is the systematic modeling of both global and individual missions, as well as all relevant mission-related information. Missions play a key role in the SoS context since they define required capabilities of constituent systems and the interactions among these systems that lead to emergent behaviors towards the accomplishment of the global mission of the SoS.

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 fulfill its own mission) without violating the independence of its constituent systems that are autonomous, acting to fulfill 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 fulfill 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 behaviors:* from the collaboration of the participating systems may emerge new behaviors. Furthermore, these behaviors may be ephemeral because the systems composing the SoS evolve independently, which may impact the availability of these behaviors. Hence, the challenge is to architect and construct a SoS in a way that emergent behaviors 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 summarized 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 fulfill 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.

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). 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 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.

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.

Keywords: Software Architecture, Architecture Description, Architecture Analysis, Safety Architecture, Cybersecurity Architecture, Mission Specification, Software-intensive Systems, Software-intensive Systems-of-Systems.

2.2 Scientific foundations

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. Architectural abstractions for formulating the architecture and re-architecture of SoS;
2. Formalism and underlying computational model to rigorously specify the architecture and re-architecture of SoS;
3. Mechanisms to construct, manage, and evolve SoSs driven by architecture descriptions, while resiliently enforcing their correctness, effectiveness, and efficiency;
4. Formalism and mechanisms for ensuring safety and cybersecurity at the architectural level and their transformations towards implementation.
5. Concepts and formalisms for specifying and operating SoS missions and generating abstract and concrete SoS architectures.

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, 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 executable models;
4. To derive/generate abstract/concrete architectural descriptions from well-defined mission specifications.

2.3 Application domains

The ArchWare Research Team develops formalisms, languages and software technologies which are transverse to application domains while providing mechanisms for customization to different architectural styles and application areas.

During 2017, addressed applications areas includes:

1. Internet-of-Things (IoT);
2. Fleet of Unmanned Aerial Vehicles (UAVs);
3. Supervisory Control And Data Acquisition (SCADA);

4. Service-oriented Robotics
5. e-Health;
6. Flood Monitoring SoS;
7. Critical SoSs.

3 Scientific achievements

3.1 The SoS Architecture Description Language (SosADL)

Keywords: Architecture Description Language (ADL), Software-intensive Systems-of-Systems (SoS).

Participants: Flavio Oquendo, Jérémy Buisson, Elena Leroux, 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 nature of systems architectures (in the sense of architectures of single systems) and systems-of-systems are very different:

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

We have continued the development of 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.

In the sequel, the main results of this line of work produced in 2017 are presented.

3.1.1 Architecturally describing the emergent behavior of SoS with SosADL

Keywords: Emergent Behavior, Architecture Description, Software Architecture, System-of-Systems (SoS), SosADL.

SoS is evolutionary developed from independent systems to achieve missions not possible to be accomplished by a single system alone. They are architecturally designed to exhibit emergent behavior, i.e. a new behavior that stem from the interactions among constituent systems, but cannot be deduced from the behaviors of the constituent

systems themselves. In this research, we developed the concepts and constructs of a novel Architecture Description Language (ADL), named SosADL, for architecturally describing emergent behaviors of software-intensive SoSs. Regarding assessment, we demonstrated the novel SosADL features for emergent behavior through an excerpt of a real application for architecting a Reconnaissance SoS, focusing on the flocking behavior of a fleet of Unmanned Aerial Vehicles (UAVs). For details see: [16].

3.1.2 Architecturally describing self-organizing SoS with SosADL

Keywords: Self-Organization, Emergence, Software Architecture, Internet-of-Things (IoT), System-of-Systems (SoS), SosADL.

A challenging issue in the architectural design of SoS for the Internet-of-Things (IoT) is how to architect an SoS in a way that the required behavior to fulfil the SoS mission will emerge. Indeed, on the one hand, at design-time, most often we do not know which are the concrete IoT systems that will become constituents of the SoS, these being predominantly identified at run-time; on the other hand, the correct architecture depends not only on the constituent systems but also, largely, on the operational environment where the SoS will be deployed. To address this challenge, this research investigated the notion of self-organization, whose mechanism makes possible that the IoT constituent systems themselves create and maintain a valid architecture enabling the production of the required emergent behavior to fulfil the SoS mission. In particular, it has showed how SosADL, a formal SoS Architecture Description Language (ADL), based on the novel π-Calculus for SoS, supports the architectural description of self-organizing SoSs for the IoT, upwardly causing SoS emergent behaviors at run-time. For details see: [17].

3.1.3 Synthesis of software architectures for SoS with SosADL

Keywords: Architecture Synthesis, Automated Constraint Solving, Architecture Description Language (ADL), Software-intensive Systems-of-Systems (SoS), SosADL.

This research addresses the issue of architecture synthesis, i.e. creating on-demand architectures automatically for a specific operational environment according to specified constraints expressed in terms of an abstract architecture. Since constituent systems are, in general, not known at design-time due to the evolving nature of SoSs, the architecture description must specify at design-time which coalitions among constituent systems are feasible at run-time. Moreover, as many SoS are being developed for safety-critical domains, additional measures must be placed to ensure the correctness and completeness of architecture descriptions. To address this issue, this research relies on SosADL, a formal language tailored for the description of SoS architectures as dynamic associations between independent constituent systems whose interactions are mediated for accomplishing a combined action. To synthesize concrete architectures that adhere to one such description, in this work we developed a formal method that systematizes the steps for producing such artifacts. The method creates an intermediate formal model which expresses the SoS architecture in terms of a constraint satisfaction problem that can be automatically analyzed for an initial set of properties. The feedback obtained in

this analysis can be used for subsequent refinements of the architecture description. A software tool was also developed to support our method by automating the generation of intermediate models and concrete architectures, thus concealing the use of constraint solvers during SoS design and development. The method and its accompanying tool were applied to model a SoS for urban river monitoring in which the feasibility of candidate abstract architectures was investigated. For details see: [3, **?**].

3.1.4 Designing software architectures of service-oriented robotic systems

Keywords: Service-Oriented Architecture (SOA), Service-Oriented Robotics, Architecture Description, Software-intensive Systems (SiS).

Robotics has experienced an increasing evolution and interest from the society in recent years. Robots are no longer produced exclusively to perform repetitive tasks in factories, they have been designed to collaborate with humans in several important application domains. Robotic systems that control these robots are, therefore, becoming larger, more complex and difficult to develop. In this scenario, Service-Oriented Architecture (SOA) has been investigated as a promising architectural style for the design of robotic systems in a flexible, reusable and productive way. Although a considerable amount of Service-Oriented Robotic Systems (SORS) has already been developed and used, most of them have been designed in an ad hoc manner. The little attention and limited support devoted to the design of SORS software architectures may not only hamper the benefits of SOA adoption but also reduce the overall quality of robotic systems, which are often used in safety-critical contexts. This research developed an Architectural Design of Service-Oriented Robotic System (ArchSORS), a process that supports a systematic design of SORS software architectures. Experimental results showed that ArchSORS can lead to software architectures of higher quality. For details see: [4].

3.1.5 Pivot model for supporting verification of SoS architectures described with SosADL

Keywords: Model Transformation, Formal Analysis, Testing, Software Architecture, Systems-of-Systems (SoS).

SosADL is a high-level architectural language used for the design of SoS architectures as well as for the description of their functional properties in terms of temporal logic. In order to verify the preservation of these properties, to validate the correct functioning of an SoS with respect to its architecture and to simulate the execution of an SoS, it is necessary to enable the access from SosADL Studio to the panel of already existing algorithms and tools used for simulation, validation and verification of software systems. Many of such tools operate on variants of labeled transition systems (LTS), including input-output Labelled Transition System (ioLTS). It is not intuitive and simple to express the behavioral semantics of an SoS architecture in terms of a labeled transition system that is a quite low-level model. Moreover, each tool used for verification is based on different kinds of transition systems which implies the redefini-

tion of the transformation. Therefore, in this research, we conceived a pivot model in our transformation chain based on a set of input-output Symbolic Transitions Systems (ioSTSs). An SoS architecture is represented by a set of ioSTS communicating to each other through unified input and output actions. Each ioSTS models the behavior of either a constituent system or a mediator.

3.1.6 Formal verification by model checking of SCADA architectures

Keywords: Formal Verification, Timed Automata, Architectural Style, Software-intensive Systems (SiS).

The design of systems for Supervisory Control And Data Acquisition (SCADA) often suffers from problems of communication and interpretation of specifications between the various designers, frequently coming from a wide range of technical fields. In order to address the architectural design of these systems, several methods have been proposed in the literature. Among them, the so-called mixed method (bottom-up/top-down), which organizes the design in two steps. In the first step (bottom-up), a model of the system is defined from a set of standardized components. This model undergoes, in the second (top-down) step, several refinements and transformations to obtain more concrete models, including code, etc. To guarantee the quality of the systems designed according to this method, in this research we developed two formal verification approaches, based on Model-Checking. The first approach concerns the verification of standardized components and allows the verification of a complete elementary control-command chain. The second one consists in verifying the model of architecture (P&ID) used for the generation of control programs.The latter is based on the definition of an architectural style in Alloy for the ANSI/ISA-5.1 standard. To support both approaches, two formal semi-automated verification flows based on Model-Driven Engineering have been proposed. This integration of formal methods in an industrial context is facilitated by the automatic generation of formal models from design models carried out by business designers. Our two approaches have been validated on a concrete industrial case of a fluid management system embedded in a ship. For details see: [2, 13, 14, 12].

3.2 Methods for architecting software-intensive systems and SoS

Keywords: Component-based Architecture, Architecture Design, Software-intensive Systems (SiS), Systems-of-Systems (SoS).

Participants: Nicolas Belloir, Isabelle Borne, Régis Fleurquin, Salah Sadou.

With the increasing complexity of Software-intensive Systems (SiS), the needs for developing methods for architecting SiSs based on component-oriented approaches has increased. It complements the research line on SoS. In the sequel, the main results of this line of work produced in 2017 are presented.

3.2.1 Preserving architectural pattern composition in component-based software architecture

Keywords: Architectural Pattern, Pattern Composition, Composable Architecture Description, Software-intensive Systems (SiS).

Component-based 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. Therefore, pattern composition has become a crucial aspect during software design. One of the shortcomings of existing work about pattern composition is the vaporization of composition information which leads to the problem of traceability and reconstructability of patterns. In this research, we proposed 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 effectiveness. By applying the approach on the composition of a set of formalized architectural patterns, including their variants, we have shown that composed patterns have become traceable and reconstructable. For details see: [**?**].

3.2.2 Automated refactoring of component-based software architecture

Keywords: Search-based Approach, Genetic Algorithm, Refactoring, Component-based Systems, Software Architecture.

During its lifecycle, a software system undergoes repeated modifications to quickly fulfill new requirements, but its underlying design is not properly adjusted after each update. This leads to the emergence of bad smells. Refactoring provides a de facto behavior-preserving approach to eliminate these anomalies. However, manually determining and performing useful refactorings is a formidable challenge, as stated in the literature. Therefore, framing object-oriented automated refactoring as a search-based technique has been proposed. However, the literature shows that search-based refactoring of component-based software has not yet received proper attention. In this research, we developed a genetic algorithm-based approach for the automated refactoring of component-based software. This approach consists of detecting component-relevant bad smells and eliminating these bad smells by searching for the best sequence of refactorings using a genetic algorithm. The developed approach consists of four steps. The first step includes studying the literature related to component-relevant bad smells and formulating bad smell detection rules. The second step involves proposing a catalog of component-relevant refactorings. The third step consists of constructing a source code model by extracting facts from the source code of a component-based software. The final step seeks to identify the best sequence of refactorings to apply to reduce the presence of bad smells in the source code model using a genetic algorithm. The latter uses bad smell detection rules as a fitness function and the catalog of refactorings as a means to explore the search space. As a case study, we conducted experiments on four real-world component-based applications. The results indicated that our approach is able to efficiently reduce the total number of bad smells by more than one half, which

is an acceptable value compared to the recent literature. Moreover, we determined that our approach is also accurate in refactoring only components suffering from bad smells while leaving the remaining components untouched whenever possible. Furthermore, a statistical analysis showed that our genetic algorithm outperforms random search and local search in terms of efficiency and accuracy on almost all the systems investigated in this work. For details see: [6].

3.2.3 Mission-based simulation for preparing SoS architecture

Keywords: Simulation, Mission, Situation Paradigm, Reaction paradigm, Software Architecture, Systems-of-Systems (SoS).

Modeling and simulation play a major role in complex system engineering. In SoS engineering, simulation helps to better understand and identify the side effects associated with the integration of autonomous constituent systems. Simulation is also a way of apprehending the emerging behaviors from this integration. The dynamic and evolving nature of the SoS environment has led us to rely on the most stable part to define them, namely their mission. In this research, we developed a simulation framework for SoS based on a conceptual model defining the mission. Mission is defined as a set of situations that require reactions. Situations are defined by rules on facts related to the SoS environment. Reactions are defined as orchestrations of services from constituent systems that must be triggered when a situation is identified. Regarding assessment, we carried out an SoS case study on health assistance. For details see: [8].

4 Software development

4.1 The SoS Architect Studio for SosADL

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

SosADL Studio, the SosADL Architecture Development Environment, is a novel environment for description, verification, simulation, and compilation/execution of SoS architectures. With SosADL Studio, SoS architectures are described using SosADL, an Architecture Description Language based on process algebra with concurrent constraints, and on a meta-model defining SoS concepts. Because constituents of an SoS are not known at design time, SosADL promotes a declarative approach of architecture families. At runtime, the SoS evolves within such a family depending on the discovery of concrete constituents. In particular, SosADL Studio enables to guarantee the correctness of SoS architectures.

At the end of 2017, the SoSADL Studio includes the following modules.

4.1.1 The type system in Coq, the type-checker and the proof generator

Participants: Jérémy Buisson.

The type-checker is based on the SoSADL type system written in Coq, which covers 2/3 of the SoSADL language. Coq proofs are generated after each successful type checking, enabling the verification of the type-checker according to the type system.

4.1.2 SosADL2Alloy: The concrete architecture generator

Participants: Milena Guessi, Flavio Oquendo.

The concrete architecture generator (SosADL2Alloy) module automatically transforms a SosADL abstract architecture into an abstract architecture in Alloy, and generates a Java class to launch a SAT solver through the Alloy Analyzer. The solutions are SoSADL concrete architectures. This module has been finished, and has to be integrated into the SosADL Studio.

4.1.3 SosADL2DEVS: The DEVS generator

Participants: Valdemar Neto, Wallace Manzano.

The SosADL2DEVS generator takes one concrete architecture as input and generates a DEVS program, which can be simulated using the MS4ME software. The simulations also generates traces. A client-server link between MS4ME and PlasmaLab – developed in the Tamis team – enables Statistical Model Checking, by reusing traces of the simulation.

4.1.4 SosADL2IoSTS: The IoSTS generator

Participants: Elena Leroux, Gersan Moguérou.

The SosADL2IoSTS generator takes one concrete architecture, and generates an ioSTS model 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. The development of a new translator from ioSTS to Uppaal has been started.

4.1.5 The SoSADL Studio

Participants: Gersan Moguérou, Jérémy Buisson, Elena Leroux, Milena Guessi, Valdemar Neto, Flavio Oquendo.

The SoSADL Studio provides an Integrated Development Environment (IDE), a simulator, a model-checker, and a statistical model-checker.

The SosADL Studio is developed under Xtext/Eclipse. It integrates the above modules into an IDE, which provides a syntactical editor to define an abstract SoS architecture, and then enable the following workflow:

- The type-checker validates the abstract SoS architecture written in SosADL, and

generates a Coq proof. This proof can be verified using the Coq proof assistant, according to the SosADL type system written in Coq.

- The concrete SoS architectures are then generated, using the SosADL2Alloy module.
- Each concrete SoS architecture can be transformed into a DEVS program, using the SosADL2DEVS module, and simulated using the MS4ME tool. The traces of the simulation enable Statistical Model Checking in PlasmaLab.
- Each concrete architectures can be transformed into ioSTS, and then into an Uppaal program, in order to verify functional properties by Model Checking.

5 Contracts and collaborations

5.1 National Initiatives

- Coordination of the GT SoS at GDR GPL (Groupement de Recherche Génie de la Programmation et du Logiciel (INS2I): GT Systems-of-Systems composed of 16 research-teams (ACADIE, ARCHWARE, CPR, DIVERSE, ESTASYS, ISC, MACAO, MAREL, MODALIS, MOVIES, RSD, SARA, SOC, SPADES, SPIRALS, TEA) from 8 UMRs (CRISTAL, I3S, IRISA, IRIT, LIRIS, LIRMM, LIX et VERIMAG), 1 UPR (LAAS), and 3 INRIA centers (Rennes Bretagne Atlantique, Lille Nord Europe, Grenoble Rhône-Alpes), the LabEx M2ST, the IRT SystemX as well as 9 engineering companies developing SoSs (AIRBUS, CAP GEMINI, CS, Naval Group, SEGULA, THALES Group, THALES Alenia Space, THALES Communications et Sécurité, THALES Recherche & Technologie) and the french association of systems engineering AFIS.

5.2 Bilateral industry grants

- SEGULA Technologies: CIFRE Scholarship

5.3 Collaborations

National Collaborations with Joint Publications:

- Flavio Oquendo has a collaboration on systems-of-systems with Khalil Drira (LAAS-CNRS)
- Salah Sadou has a collaboration on reuse of architectural constraints with Chouki Tibermancine and Christophe Dony (LIRMM)

International Collaborations with Joint PhD Supervision:

- Flavio Oquendo:

  - USP - University of Sao Paulo - ICMC Research Institute, Sao Carlos, Brazil (Elisa Nakagawa)
  - UFRN - Federal University of Rio Grande do Norte, Natal, Brazil (Thais Batista)

- Salah Sadou:
  - University of Science and Technology of Houari Boumedienne, Alger, Algeria (Mohamed Ahmed Nacer)

- Isabelle Borne:
  - LISCO, University Badji Mokhtar Annaba, Algeria (Djamel Meslati)

Local Collaborations:

6 Dissemination

6.1 Promoting scientific activities

Research and Doctoral Supervizing Awards (PEDR)

- Flavio Oquendo: PEDR (2016-2020)
- Salah Sadou: PEDR (2014-2017)

Chair/Member of Conference Steering Committees

- Flavio Oquendo:
  - European Conference on Software Architecture - ECSA (Steering Committee Chair)
  - IEEE International Conference on Software Architecture - ICSA (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)

- Salah Sadou:
  - CIEL: French Conference on Software Engineering (Steering Committee Member)

Chair/Member of Conference Program Committees

- Nicolas Belloir:
  - AICCSA: ACS/IEEE International Conference on Computer Systems and Applications, Conference Track on Software Engineering, 2017
  - INFORSID: INFormatique des ORganisations et Systémes d'Information et de Décision, 2017
- Isabelle Borne:
  - SESOS: ACM/IEEE ICSE International Workshop on Software Engineering for Systems-of-Systems, 2017
  - AICCSA: ACS/IEEE International Conference on Computer Systems and Applications, Conference Track on Software Engineering, 2017 (PC Chair)
- Jérémy Buisson:
  - ICCS: International Conference on Computational Science, 2017
  - AICCSA: ACS/IEEE International Conference on Computer Systems and Applications, Conference Track on Software Engineering, 2017
- Régis Fleurquin
  - AICCSA: ACS/IEEE International Conference on Computer Systems and Applications, Conference Track on Software Engineering, 2017
- Flavio Oquendo:
  - SOSE: IEEE International Conference on System-of-Systems Engineering, 2017 (PC Chair of Special track on Software-intensive Systems-of-Systems)
  - SISOS: ACM International Symposium On Applied Computing Program, Conference Track on Software Software-intensive Systems-of-Systems, 2017 (PC Chair)
  - SESOS: ACM/IEEE ICSE International Workshop on Software Engineering for Systems-of-Systems, 2017 (PC Chair)
  - ICSA: IEEE International Conference on Software Architecture, 2017
  - ECSA: European Conference on Software Architecture, 2017
  - SATTA: ACM Symposium On Applied Computing Program, Conference Track on Software Architecture: Theory, Technology, and Applications, 2017
  - CYBER: International Conference on Cyber-Technologies and Cyber-Systems, 2017
  - ICSEA: International Conference on Software Engineering Advances, 2017
  - SOFTENG: International Conference on Advances and Trends in Software Engineering, 2017
  - ICSOFT: International Conference on Software and Data Technologies, 2017

  - ICAS: International Conference on Autonomic and Autonomous Systems, 2017
  - ICONS: International Conference on Systems, 2017
  - COMPLEXIS: International Conference on Complexity, 2017
  - CAL: French Conference on Software Architecture, 2017
  - SBES: Brazilian Symposium on Software Engineering, 2017

- Salah Sadou:
  - AICCSA: ACS/IEEE International Conference on Computer Systems and Applications, Conference Track on Software Engineering, 2017 (PC Chair)
  - SESOS: ACM/IEEE ICSE International Workshop on Software Engineering for Systems-of-Systems, 2017

6.1.1 Journal

Member of the Editorial Boards

- Flavio Oquendo:
  - Springer Journal of Software Engineering Research and Development (Member of the Editorial Board)

6.1.2 Scientific Expertise

- Flavio Oquendo:
  - Scientific Expert acting as reviewer and evaluator of R&D Projects for the European Commission (Horizon H2020)
  - Expert acting as evaluator of R&D Projects for the ANR (Agence Nationale de la Recherche) on Software Sciences and Technologies
  - Expert acting as evaluator of R&D Projects for the FWO (Research Foundation Flanders, Belgium) on Cyber-Physical Systems and SoS

6.1.3 Laboratory Administration

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

6.1.4 Academic Council (CAC)

- Salah Sadou: Member of the CAC (Commission recherche du conseil académique) of UBS

6.2 Teaching

6.2.1 Teaching

- Academics of ARCHWARE teach at the Research Master on Computer Science of Université Bretagne Sud

6.2.2 Teaching Responsibility

- Salah Sadou: Head of the Engineering Degree on Software Cybersecurity of ENSIBS School of Engineering
- Flavio Oquendo: Head of the Research Master Degree on Computing of Université Bretagne Sud (part of the regional research master in Computer Science headed by Université de Rennes 1)

Books and Monographs

[1] *Proceedings of the 1st ACM SAC Conference Track on Software-intensive Systems-of-Systems (SiSoS 2017)*, Marrakesh, Morocco, ACM, April 2017, https://hal.archives-ouvertes.fr/hal-01445350.

Doctoral dissertations and "Habilitation" theses

[2] S. M. Kesraoui, *Integration of formal verification techniques into a control-command system design approach : application to SCADA architectures*, Theses, Université de Bretagne Sud, May 2017, https://tel.archives-ouvertes.fr/tel-01738049.

[3] M. G. Margarido, *Synthesis of software architectures for systems-of-systems : an automated method by constraint solving*, Theses, Université de Bretagne Sud, September 2017, https://tel.archives-ouvertes.fr/tel-01793110.

Articles in referred journals and book chapters

[4] L. Bueno Ruas De Oliveira, E. Leroux, K. Romero Felizardo, F. Oquendo, E. Yumi Nakagawa, "ArchSORS: A Software Process for Designing Software Architectures of Service-Oriented Robotic Systems", *The Computer Journal 60*, 9, September 2017, p. 1363–1381, https://hal.archives-ouvertes.fr/hal-01442597.

[5] V. V. Graciano Neto, C. E. Barros Paes, L. Garcés, M. Guessi, W. Manzano, F. Oquendo, E. Y. Nakagawa, "Stimuli-SoS: a model-based approach to derive stimuli generators for simulations of systems-of-systems software architectures", *Journal of the Brazilian Computer Society 23*, 1, December 2017, https://hal.archives-ouvertes.fr/hal-02132109.

[6] S. Kebir, I. Borne, D. Meslati, "A Genetic Algorithm-Based Approach for Automated Refactoring of Component-Based Software", *Information and Software Technology 88*, August 2017, p. 17 – 36, https://hal.archives-ouvertes.fr/hal-01705479.

[7] M. T. Thon That, S. Sadou, F. Oquendo, I. Borne, "Preserving architectural pattern composition information through explicit merging operators", *Future Generation Computer Systems 47*, June 2017, p. 97–112, https://hal.archives-ouvertes.fr/hal-01102209.

Publications in Conferences and Workshops

[8] R. Benabidallah, I. Cherfa, S. Sadou, M. Ahmed-Nacer, "Situation/Reaction Paradigm for SoS Simulation", *in : 26th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2017, Poznan, Poland, June 21-23, 2017*, S. Reddy, W. Cellary, M. Fugini (editors), IEEE Computer Society, p. 48–53, 2017, https://doi.org/10.1109/WETICE.2017.25.

[9] I. Borne, M. H. Fendali, D. Meslati, "Understanding evolution in systems of systems", *in : 2017 IEEE International Systems Engineering Symposium (ISSE)*, IEEE, Vienna, Austria, October 2017, https://hal.archives-ouvertes.fr/hal-01705414.

[10] L. Garcés, F. Oquendo, E. Yumi Nakagawa, " A Process to Establish, Model and Validate Missions of Systems-of-Systems in Reference Architectures.", *in : The 32nd ACM Symposium on Applied Computing*, p. 1–8, Marrakesh, Morocco, April 2017, https://hal.archives-ouvertes.fr/hal-01429108.

[11] J. Leite, T. Batista, F. Oquendo, "Architecting IoT Applications with SysADL", *in : 2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*, IEEE, p. 92–99, Gothenburg, France, April 2017, https://hal.archives-ouvertes.fr/hal-02132111.

[12] S. Mesli, A. Bignon, D. Kesraoui, A. Toguyeni, F. Oquendo, P. Berruet, "Vérification formelle de chaines de contrôle-commande d'éléments de conception standardisés", *in : Proceedings of the 11th International Conference on Modeling, Optimization & Simulation (MOSIM 2016)*, Montréal, Canada, August 2016, https://hal.archives-ouvertes.fr/hal-01441589.

[13] S. Mesli, D. Kesraoui, F. Oquendo, A. Bignon, A. Toguyéni, P. Berruet, "Formal Verification of Software-Intensive Systems Architectures Described with Piping and Instrumentation Diagrams ", *in : Proceedings of the 10th European Conference on Software Architecture (ECSA 2016)*, *LNCS*, 9839, Springer, p. 210–226, Copenhagen, Denmark, November 2016, https://hal.archives-ouvertes.fr/hal-01440744.

[14] S. Mesli, A. Toguyéni, A. Bignon, F. Oquendo, D. Kesraoui, P. Berruet, "Formal and Joint Verification of Control Programs and Supervision Interfaces for Sociotechnical Systems Components", *in : Proceedings of the 13th IFAC Symposium on Analysis, Design, and Evaluation of Human-Machine Systems (HMS 2016)*, IFAC, p. 427–467, Kyoto, Japan, August 2016, https://hal.archives-ouvertes.fr/hal-01441587.

[15] E. Y. Nakagawa, F. Oquendo, P. Avgeriou, R. Santos, "Proceedings of the IEEE/ACM Joint 5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems", *in : 2017 IEEE/ACM Joint 5th International Workshop on Software Engineering for Systems-of-Systems and 11th Workshop on Distributed Software Development, Software Ecosystems and Systems-of-Systems (JSOS)*, IEEE, Buenos Aires, Argentina, May 2017, https://hal.archives-ouvertes.fr/hal-02132114.

[16] F. Oquendo, "Architecturally describing the emergent behavior of software-intensive system-of-systems with SosADL", *in : 2017 12th System of Systems Engineering Conference (SoSE)*, IEEE, p. 1–6, Waikoloa, United States, June 2017, `https://hal.archives-ouvertes.fr/hal-02132117`.

[17] F. Oquendo, "Software architecture of self-organizing systems-of-systems for the Internet-of-Things with SosADL", *in : 2017 12th System of Systems Engineering Conference (SoSE)*, IEEE, p. 1–6, Waikoloa, United States, June 2017, `https://hal.archives-ouvertes.fr/hal-02132121`.