Activity Report 2018

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
Vanea Chiprianov, Assistant Professor, Université Bretagne Sud
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
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 2018, addressed applications areas includes:

1. Internet-of-Things (IoT);
2. Fleet of Unmanned Aerial Vehicles (UAVs);
3. Crowd Management SoS;

4. Smart e-Health;
5. Flood Monitoring SoS;
6. 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 run-time 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 2018 are presented.

3.1.1 Enhancing SosADL with support for emergent behavior

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

A major research challenge for the design of a software-intensive SoSs is the formal description of its software architecture. The main complicating factor is that an SoS, by definition, has its architecture designed to produce emergent behavior for fulling missions. But, by definition, an emergent behavior is an "unexpected" behavior. By "unexpected", we mean a behavior of a whole (i.e. the SoS) that cannot be predicted through analysis only of its parts (i.e. the constituent systems of the SoS), or stated simply: the behavior of the SoS is other than the sum of the behaviors of its constituent systems. The oxymoron is thereby: how to formally design and describe the software architecture of an SoS to exhibit "expected" (by design) "unexpected" (emergent) behaviors that stem from the interactions among the constituent systems of the SoS? To address this oxymoron, this research investigated the notion of emergent behavior for clarifying the boundaries of this notion for SoS and developed an architectural emergentist framework to enable the architectural description of emergent behavior of software-intensive SoS based on the supervenience principle. In our framework, an emergent behavior is a macro-scale behavior upwardly caused by a set of micro-scale behaviors according to alternative approaches: (i) endogenous, when the micro-scale behaviors originate from inside constituent systems of the SoS; (ii)exogenous, when oppositely the micro-scale behaviors originate from outside constituent systems. For assessing the emergentist framework, a real application was carried out for architecting a Reconnaissance SoS, focusing on the flocking behavior of a fleet of Unmanned Aerial Vehicles (UAVs). For details see: [18, 16].

3.1.2 Mission-driven design of SoS architectures described with SosADL

Keywords: Mission Modeling, Semi-Automated Architecture Design, Software Architecture, Systems-of-Systems, SosADL.

The formulation of missions is the starting point to the development of SoSs, being used as a basis for the specification, verification and validation of SoS architectures. Specifying, verifying and validating architectural models for SoS are complex tasks compared to usual systems, the inner complexity of SoS relying specially on emergent behaviors, i.e. features that emerge from the interactions among constituent parts of the SoS which cannot be predicted even if all the behaviors of all parts are completely known. In this research, we addressed the synergetic relationship between missions and architectures of software-intensive SoS, giving a special attention to emergent behaviors which are created for achieving formulated missions. We developed a design approach for the architectural modeling of SoS driven by the mission models. In our proposal, the mission model is used to both derive, verify and validate SoS architectures. As first step, we define a formalized mission model, then we generate the structure of the SoS architecture by applying model transformations. Later, when the architect specifies the behavioral aspects of the SoS, we generate concrete SoS architectures that will be verified and validated using simulation-based approaches, in particular regarding emergent behaviors. The verification uses statistical model checking to verify whether specified properties are satisfied, within a degree of confidence. The formalization in terms of a temporal logic and statistical model checking are the formal foundations of the developed approach. A toolset that implements the whole approach was also developed and experimented. For details see: [2].

3.1.3 Simulation-driven design of SoS architectures described with SosADL

Keywords: Simulation, Model-Based Engineering, Software Architecture, Systems-of-Systems, SosADL.

Correct SoS operations depend on a precise specification of the SoS structure and a rigorous assessment of its behaviors. However, besides limitations on languages to jointly capture SoS structure and behavior, predictions on the SoS emergent behaviors rely on constituent systems not totally known at design-time. To address this issue, in this research we developed solutions founded on a formal architectural description language to support an early evaluation of SoS behaviors regarding its inherent SoS structure and dynamics through simulations. The outcomes of this research comprises (i) a model transformation approach for automatically producing simulation models from SoS software architecture descriptions, combining SoS structure and behavior description in a same solution, (ii) an SoS software architecture evaluation method for SoS operation prediction considering the inherent changes that can occur, (iii) environment modeling and automatic generation of stimuli generators to sustain the SoS simulation, delivering data to feed such simulation, and (iv) a method for the automatic synchronization between the runtime descriptive architecture (changed at runtime due to dynamic architecture) and its original prescriptive architecture based on model discovery and recovery mechanisms and a backward model transformation. We conducted case studies to assess the proposed approach using Flood Monitoring SoS and Space SoS. The assessment showed a high accuracy to (i) produce fault-free and operational simulations for SoS software architectures, (ii) support a reliable evaluation and prediction of SoS operation at design-time, (iii) automatically generate stimuli generators to sustain and feed the simulation execution, and (iv) maintain the synchronization between descriptive and prescriptive versions of the SoS architecture. For details see: [3, 12].

3.1.4 Enhancing the implementation of SosADL by the automatic transformation from Ecore metamodels towards Gallina inductive types

Keywords: Model Transformation, QVT, Ecore, Xtext, Coq, SosADL.

When engineering a language (and its compiler), it is convenient to use widespread and easy-to-use Model-Driven Engineering frameworks like Xtext that automatically generate a compiler infrastructure, and even a full-featured Integrated Development Environment (IDE). At the same time, a formal workbench such as a proof assistant is helpful to ensure the language specification is sound. Unfortunately, the two technical spaces hardly integrate. In this research, we developed a transformation from Ecore's metametamodel to Coq's language named Gallina/Vernacular. The structural fragment of Ecore is fully handled. At the cost of not being bijective, our transformation has relaxed constraints over the input metamodel, in comparison to previous state of the art. To validate, we have used the proposed transformation to implement a proof-carrying code type checker for the SosADL language. For details see: [8].

3.1.5 Verification of SoS architectures described with SosADL

Keywords: Testing, Model Checking, Software Architecture, Systems-of-Systems, SosADL.

There are at least three important aspects that should be addressed while testing

SoS: unit testing of the constituent systems, integration and regression testing of SoS. This research addresses the integration testing under two assumptions: (1) final SoS is a two-layer system, and (2) each individual (constituent) system reflects the quality expectations towards the SoS (have been already tested individually). A major challenge of this research is to determine how to design a test suite that will check that the SoS mission is achieved. One of the ways is to derive a model (graph, transition system, etc.) from the formal description of the SoS mission. Based on this model and using one of the known, adapted or newly proposed test generation technique, we will obtain a set of abstract test cases accompanied by test verdicts. Finally, each abstract test cases should be transformed into sequences of stimuli, understandable by the SoS under test, which will guide SoS thought the test. Complementary to testing, to ensure that an SoS architecture is well-behaved, this research also investigates the application of model checking with UPPAAL. SosADL is already supported by a semantical model expressed as input-output Symbolic Transitions Systems (ioSTS). The set of ioSTS representing an SoS architecture is transformed into a set of Timed Automata (TA) that communicate using input, output actions and that exchange their data through global variables. This set of TA forms a Network of Timed Automata (NTA) which will be further developed to support formal verification by model checking.

3.1.6 Reference architecture for healthcare SoS and its description with SosADL

Keywords: Reference Architecture, Healthcare Supportive SoS, Software Architecture, System-of-Systems, SosADL.

Population ageing has been taking place all over the world, being estimated that 2.1 billion people will be aged 60 or over in 2050. Healthcare Supportive Home (HSH) SoSs have been proposed to overcome the high demand of remote home care for assisting an increasing number of elderly people living alone. Since a heterogeneous team of healthcare professionals need to collaborate to continually monitor health status of chronic patients, a cooperation of pre-existing e-Health systems, both outside and inside home, is required. However, current HSH solutions are proprietary, monolithic, high coupled, and expensive, and most of them do not consider their interoperation neither with distributed and external e-Health systems, nor with systems running inside the home (e.g., companion robots or activity monitors). These systems are sometimes designed based on local legislations, specific health system configurations (e.g., public, private or mixed), care plan protocols, and technological settings available; therefore, their reusability in other contexts is sometimes limited. As a consequence, these systems provide a limited view of patient health status, are difficult to evolve regarding the evolution of patient's health profile, do not allow continuous patients monitoring, and present limitations to support the self-management of multiple chronic conditions. To contribute to solve the aforementioned challenges, this research establishes a reference architecture for supporting the development of quality HSH SoSs. Indeed, we consider HSH as SoS instead of a large monolith, which achieve their missions (e.g., improvement of patients' quality of life) through the behavior that emerges as result of collaborations among their constituents. To establish this reference architecture, a systematic process was adopted. As a result, this reference architecture comprises domain knowledge and architectural solutions (i.e., architectural patterns and tactics) described using conceptual, mission, and quality architectural viewpoints. To assess the proposed reference architecture, a case study was performed by instantiating it to design the software architecture of a HSH SoS to assist at home patients suffering of diabetes mellitus. Results evidenced the the developed reference architecture is viable as well as enables to guide the development of reusable, interoperable, reliable, secure, and adaptive HSH SoSs. For details see: [3].

3.2 Methods for architecting SoSs

Keywords: Cybersecurity, Secure Architecture Design, Architecture Description, Software-intensive Systems-of-Systems (SoS).

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

With the growing complexity of SoS architectures, the needs for developing methods for architecting SoSs has increased. In the sequel, the main results of this line of work produced in 2018 are presented.

3.2.1 Method for the evolutionary development of SoSs

Keywords: Dynamic Reconfiguration, Reconfiguration Pattern, SoS Architecture, Systems-of-Systems.

SoSs are mostly concurrent, widely distributed, and inherently composed of independent systems. SoSs evolve in unpredictable environments and are constantly integrating newly identified systems. In this research, we deal with the problem of the evolutionary development of an SoS by using dynamic reconfiguration. We defined a process for developing configuration models and a reconfiguration design process incorporating the concept of reconfiguration pattern. For assessing the validity and feasibility of the developed approach, we have conducted an experimental framework based on a real case study of a French emergency service. For details see: [5].

3.2.2 Method for involving domain experts in the development of SoSs

Keywords: SoS Requirements, SoS Architecture, Requirements-to-Architecture, Systems-of-Systems.

An operational environment of an SoS is in perpetual evolution thus forcing a recurrent adaptation of the concerned SoS. This yet worst in case of poor communication between the requirement definition phase and the design phase. In this research, we developed a method for addressing SoS engineering using the concepts Mission and Role. The former allows the definition of the SoS behavior, while the later allows to abstract this definition with respect to the constituent systems that may actually exist in the environment. This definition is translated into an abstract architecture, which serves as a guide and controller of the choices proposed by the system architect during the design and evolution phases. With our approach we have correctly defined an SoS concerning crowd management during a sport event. For details see: [9].

3.2.3 Method for analyzing security in the development of SoSs in the IoT

Keywords: Cybersecurity, Internet-of-Things, SoS Architecture, Systems-of-Systems.

The control and protection of user data is a key aspect in the design and deployment of SoSs in the IoT. In this research, we developed a security-based modelling language for IoT SoSs, which explicitly represents data access controls. The language leverages the analysis of potential security failures resulting from a series of interactions between heterogeneous constituents of an SoS. We implemented a tool that automatically transforms IoT models into BIP models, which can then be simulated and analyzed for security guarantees. We illustrate the features of our language with a use-case inspired by an industrial scenario. For details see: [7].

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. For details see: [15].

At the end of 2018, 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: Generating Concrete SoS Architectures based on SosADL

Participants: Milena Guessi, Flavio Oquendo, Gersan Moguérou,.

The concrete architecture generator (SosADL2Alloy) module automatically transforms an abstract SoS architecture into an abstract architecture in Alloy, and generates a Java class to launch a SAT solver through the Alloy Analyzer. The solutions are concrete SoS architectures. During the integration of this module into the SosADL Studio, we have improved the quality of the generated concrete SoS architectures.

4.1.3 SosADL2DEVS: Generating and Simulating Concrete Architectures

Participants: Valdemar Neto, Wallace Manzano, Gersan Moguérou.

The SosADL2DEVS generator takes one concrete architecture as input and generates a DEVS program, which can be simulated using the MS4ME simulation tool. The simulations generate traces. A client-server link between MS4ME and PlasmaLab enables Statistical Model Checking, by reusing traces of the simulation. The SosADL2DEVS module now generates DEVS programs, which can evolve dynamically during a simulation inside MS4ME.

Although some problems remain in this module to be perfectly generic and well integrated in the SosADL Studio, this module is now available in the SosADL Studio. As this module is written in Xtend/Java under Xtext/Eclipse, it can run on any platform. Regarding DEVS, in 2018, only the Windows version of MS4ME was available.

4.1.4 SosADL2IoSTS: The SosADL Support for Verification

Participants: Elena Leroux, Gersan Moguérou.

The SosADL2IoSTS generator takes one concrete SoS architecture, and generates an ioSTS model in order to verify functional properties of SoS. The development of the translator from ioSTS to Uppaal NTA is ongoing work.

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

- Public-private collaboration on the cybersecurity of large public events between UBS, ENGIE and other companies in the domain of the cybersecurity of systems-of-systems. This ongoing collaboration aims to support the design and operation of large-scale sociotechnical systems-of-systems in open environments. It, in particular, aims to support the 2024 Summer Olympics in Paris. The ARCHWARE team brings to this joint R&D project its expertise on security by design for mastering emergent behaviors in sociotechnical systems-of-systems architectures.
- 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:

- Nicolas Belloir collaborates with IMT Atlantique (PASS research-team) and the Chaire de Cyberdefence des Systemes Naval;
- Salah Sadou leads a project on Cybersecurity with 6 Post-Docs funded by the Brittany council and the UBS. The Post-Docs are allocated in 5 laboratories of the UBS in different related disciplines.

6 Dissemination

6.1 Promoting scientific activities

Research and Doctoral Supervizing Awards (PEDR)

- Flavio Oquendo: PEDR (2016-2020)

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)
  - Workshop on Distributed Development of Software, Ecosystems and Systems-of-Systems - WDES (Steering Committee Member)

- 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, 2018
  - ICAASE: International Conference on Advanced Aspects of Software Engineering, 2018
- Isabelle Borne:
  - SESOS: ACM/IEEE ICSE International Workshop on Software Engineering for Systems-of-Systems, 2018
- Jérémy Buisson:
  - ICCS: International Conference on Computational Science, 2018
- Régis Fleurquin
  - AICCSA: ACS/IEEE International Conference on Computer Systems and Applications, Conference Track on Software Engineering, 2018
- Flavio Oquendo:
  - SOSE: IEEE International Conference on System-of-Systems Engineering, 2018 (PC Chair of Special track on Software-intensive Systems-of-Systems)
  - ISIVC: International Symposium on Signal, Image, Video and Communications, 2018 (PC Chair)
  - SISOS: ACM International Symposium On Applied Computing Program, Conference Track on Software Software-intensive Systems-of-Systems, 2018 (PC Chair)
  - SESOS: ACM/IEEE ICSE International Workshop on Software Engineering for Systems-of-Systems, 2018 (PC Chair)

  - ICSA: IEEE International Conference on Software Architecture, 2018
  - ECSA: European Conference on Software Architecture, 2018
  - CPSIOT: International Conference on Cyber Physical Systems and IoT, 2018
  - CYBER: International Conference on Cyber-Technologies and Cyber-Systems, 2018
  - ICSEA: International Conference on Software Engineering Advances, 2018
  - SOFTENG: International Conference on Advances and Trends in Software Engineering, 2018
  - ICSOFT: International Conference on Software and Data Technologies, 2018
  - ICAS: International Conference on Autonomic and Autonomous Systems, 2018
  - ICONS: International Conference on Systems, 2018
  - COMPLEXIS: International Conference on Complexity, 2018
  - ICT4I40: IEEE ISCC International Workshop on Information and Communication Technologies for Industry 4.0, 2018
  - SEET: ICSE Track on Software Engineering Education and Training, 2018
  - CAL: French Conference on Software Architecture, 2018
  - SBES: Brazilian Symposium on Software Engineering, 2018
  - GE: ACM/IEEE ICSE Workshop on Gender Equality in Software Engineering, 2018

- Salah Sadou:
  - SESOS: ACM/IEEE ICSE International Workshop on Software Engineering for Systems-of-Systems, 2018

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 Trustworthy Software-intensive Systems

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 6th ACM/IEEE International Workshop on Software Engineering for Systems-of-Systems (SESoS 2018) at the 40th International Conference on Software Engineering (ICSE 2018)*, Gothenburg, Sweden, ACM, May 2018, `https://hal.archives-ouvertes.fr/hal-02570260`.

Doctoral dissertations and "Habilitation" theses

[2] E. Ferreira silva, *Mission-driven Software-intensive System-of-Systems Architecture Design*, Theses, Université de Bretagne Sud ; Universidade federal do Rio Grande do Norte (Natal, Brésil), December 2018, `https://tel.archives-ouvertes.fr/tel-02372206`.

[3] L. M. Garcès Rodriguez, *A reference architecture for healthcare supportive home systems from a systems-of-systems perspective*, Theses, Université de Bretagne Sud ; Universidade de São Paulo (Brésil), May 2018, `https://tel.archives-ouvertes.fr/tel-02089352`.

[4] V. V. Graciano Neto, *A simulation-driven model-based approach for designing softwareintensive systems-of-systems architectures*, Theses, Université de Bretagne Sud ; Universidade de São Paulo (Brésil), March 2018, `https://tel.archives-ouvertes.fr/tel-02146340`.

[5] F. Petitdemange, *Evolutionary development of systems of systems with a dynamic reconfiguration pattern approach*, Theses, Université de Bretagne Sud, December 2018, https://tel.archives-ouvertes.fr/tel-02073913.

Publications in Conferences and Workshops

[6] T. Batista, F. Oquendo, J. Leite, “Modeling and Executing Software Architecture Using SysADL”, *in: 2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*, IEEE, Seattle, United States, April 2018, https://hal.archives-ouvertes.fr/hal-02132141.

[7] D. Beaulaton, N. Ben Said, I. Cristescu, R. Fleurquin, A. Legay, J. Quilbeuf, S. Sadou, “A Language for Analyzing Security of IOT Systems”, *in: SoSE 2018 - 13th Annual Conference on System of Systems Engineering*, IEEE, p. 37–44, Paris, France, June 2018, https://hal.inria.fr/hal-01960860.

[8] J. Buisson, S. Rehab, “Automatic Transformation from Ecore Metamodels towards Gallina Inductive Types”, *in: MODELSWARD 2018*, Santa Cruz, Portugal, January 2018, https://hal.archives-ouvertes.fr/hal-01693939.

[9] I. Cherfa, S. Sadou, N. Belloir, R. Fleurquin, D. Bennouar, “Involving the Application Domain Expert in the Construction of Systems of Systems”, *in: IEEE - 13th System of Systems Engineering Conference (SoSE 2018)*, Paris, France, June 2018, https://hal.archives-ouvertes.fr/hal-01978238.

[10] K. Drira, F. Oquendo, A. Legay, T. Batista, “Editorial Message Track on Software-intensive Systems-of-Systems (SiSoS) of the 33rd ACM/SIGAPP Symposium On Applied Computing (SAC 2018)”, *in: SAC 2018 - The 33rd ACM/SIGAPP Symposium On Applied Computing*, p. 1–3, Pau, France, April 2018, https://hal.laas.fr/hal-01666389.

[11] L. Garcès, F. Oquendo, E. Y. Nakagawa, “Towards a Taxonomy of Software Mediators for Systems-of-Systems”, *in: The VII Brazilian Symposium on Software Components, Architectures, and Reuse*, ACM Press, p. 53–62, Sao Carlos, Brazil, September 2018, https://hal.archives-ouvertes.fr/hal-02132149.

[12] V. V. Graciano Neto, L. Garcès, M. Guessi, C. E. d. B. Paes, F. Oquendo, W. Manzano, E. Y. Nakagawa, “ASAS: An Approach to Support Simulation of Smart Systems”, *in: Hawaii International Conference on System Sciences (HICSS 2018)*, Waikoloa, Hawaii, United States, January 2018, https://hal.archives-ouvertes.fr/hal-02570228.

[13] M. L. Kerdoudi, M. L. Kerdoudi, C. Tibermacine, S. Sadou, “Spotlighting Use Case Specific Architectures”, *in: ECSA: European Conference on Software Architecture, LNCS*, 11048, p. 236–244, Madrid, Spain, September 2018, https://hal-lirmm.ccsd.cnrs.fr/lirmm-02124337.

[14] J. Leite, T. Batista, F. Oquendo, E. Silva, L. Santos, V. Cortez, “Designing and Executing Software Architectures Models Using SysADL Studio”, *in: 2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*, IEEE, Seattle, United States, April 2018, https://hal.archives-ouvertes.fr/hal-02132146.

[15] F. Oquendo, J. Buisson, E. Leroux, G. Moguérou, “A Formal Approach for Architecting Software-intensive Systems-of-Systems with Guarantees”, *in: 13th Annual Conference on System of Systems Engineering (SoSE)*, Paris, France, July 2018, https://hal.archives-ouvertes.fr/hal-01999617.

[16] F. Oquendo, "Exogenously Describing Architectural Emergent Behaviors of Systems-of-Systems with SosADL", *in: 2018 13th Annual Conference on System of Systems Engineering (SoSE)*, IEEE, p. 268–275, Paris, France, June 2018, https://hal.archives-ouvertes.fr/hal-02132153.

[17] F. Oquendo, "Formally Describing Self-organizing Architectures for Systems-of-Systems on the Internet-of-Things", *in: 12th European Conference on Software Architecture, ECSA 2018*, Springer, p. 20–36, Madrid, Spain, September 2018, https://hal.archives-ouvertes.fr/hal-02132137.

[18] F. Oquendo, "On the Emergent Behavior Oxymoron of System-of-Systems Architecture Description", *in: 2018 13th Annual Conference on System of Systems Engineering (SoSE)*, IEEE, p. 417–424, Paris, France, June 2018, https://hal.archives-ouvertes.fr/hal-02132155.

[19] F. Petitdemange, I. Borne, J. Buisson, "Modeling System of Systems configurations", *in: 2018 13th Annual Conference on System of Systems Engineering (SoSE)*, IEEE, p. 392–399, Paris, France, June 2018, https://hal.archives-ouvertes.fr/hal-02021350.