Activity Report 2019

Research-Team ArchWare

Software Architecture

Architecting Software-intensive Systems and Systems-of-Systems

D4 – Language and Software Engineering

Contents

1 Team composition **1**

2 Overall objectives **2**

2.1 Overview 2

2.2 Scientific foundations 5

2.3 Application domains 5

3 Scientific achievements **6**

3.1 The SoS Architecture Description Language (SosADL) 6

3.1.1 Enhancing SosADL with software mediators as first-class entities of SoS architectures 7

3.1.2 Enhancing SosADL with support for mediators extended with broadcast communication 8

3.1.3 Enhancing SosADL with support for architecting exogenous SoSs 8

3.1.4 Enhancing SosADL with support for coping with uncertainty in SoSs 9

3.1.5 Further enhancing SosADL with Digital Twins for coping with uncertainty in SoSs 9

3.1.6 Enhancing the implementation of SosADL by bridging Ecore and Coq: SosADL Type-Checker with Proof-Carrying Code . . . 10

3.2 Supporting methods for architecting secure SoSs 10

3.2.1 Designing SoS architectures from mission definition 11

3.2.2 Developing secure SoSs for rapid deployment 11

3.2.3 Security analysis of SoSs in the IoT using attack trees 11

3.2.4 Developing SoSs for the the battlefield: a proof-of-concept to deal with digitalization 12

4 Software development **12**

4.1 The SoS Architect Studio for SosADL 12

4.1.1 The type system in Coq, the type-checker and the proof generator 13

4.1.2 SosADL2Alloy: Generating concrete SoS architectures based on SosADL 13

4.1.3 SosADL2DEVS: Generating and simulating concrete architectures 13

4.1.4 SosADL2IoSTS: The SosADL support for architecture verification 13

4.1.5 The SoSADL Studio 14

5 Contracts and collaborations **14**

5.1 National Initiatives 14
5.2 Bilateral industry grants 15
5.3 Collaborations 15

6 Dissemination **16**
6.1 Promoting scientific activities 16
6.1.1 Journal 18
6.1.2 Scientific Expertise 18
6.1.3 Laboratory Administration 18
6.1.4 Academic Council (CAC) 18
6.2 Teaching 18
6.2.1 Teaching 18
6.2.2 Teaching Responsibility 19

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
Jamal El Hachem, 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

Associate members (IRISA/SEGULA Agreement)

Olga Goubali-Rodier, PhD, SEGULA Technologies - R&D Division
Soraya Mesli-Kesraoui, PhD, SEGULA Technologies - R&D Division

Research engineers

Gersan Moguérou, Research Engineer, Université Bretagne Sud

PhD students

Delphine Beaulaton
Raounak Benabidallah
Rymel Benabidallah
Nathalie Bouldoukian
Youcef Bouziane
Elyes Cherfa
Imane Cherfa
Elia Christy-Fikany
Nan Zhang Messe
Paul Perrotin

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 2019, addressed applications areas includes:

1. Internet-of-Things (IoT), Industrial Internet-of-Things (IIoT), Internet-of-Mobile Things (IoMT), Internet-of-Vehicles (IoV);
2. Intelligent Transportation Systems;

3. Smart e-Health;
4. Battlefield Engineering;
5. 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 2019 are presented.

3.1.1 Enhancing SosADL with software mediators as first-class entities of SoS architectures

Keywords: Mediator, Connector, Software Architecture, Systems-of-Systems.

This research studied the issue of mediators as first-class software entities to be applied in the construction of SoS architectures. The following four steps were conducted in this research: (i) identification of mediation requirements to enable SoS properties; (ii) establishment and categorization of twelve types of mediators, for enabling capabilities of communication and control of constituent systems interactions and conversion of heterogeneous messages exchanged through a mediation infrastructure; (ii) specification of duties, behaviors, assumptions, and guarantees of mediators; and (iv) organization of mediators in three layers, namely, the constituents and consumer systems layer; the

communication, conversion and coordination layer; and the control layer. The proposed kinds of mediators were applied as the backbone for constructing SoS architectures in different application fields, namely, flood monitoring system-of-systems and health-care supportive home system-of-systems. In fine, the proposed enhancements for mediators suitably support the integration of independent constituent systems, provide strategies to manage emergent behaviors, define different schemes of control authorities, offer elements to support SoS evolution, and promote the resilience and adaptability of SoS architectures. For details see: [6].

3.1.2 Enhancing SosADL with support for mediators extended with broadcast communication

Keywords: Broadcasting Communication, Architecture Description Language (ADL), Software-intensive Systems-of-Systems (SoS), SosADL, Flood Monitoring System-of-Systems, Health-Care Supportive Home System-of-Systems.

An SoS is architecturally designed to exhibit emergent behavior from the interactions among independent constituent systems. With the upcoming generation of self-driving vehicles, an important case of emergent behavior is vehicle platooning. In a platoon, a group of vehicles (which is dynamically formed) safely travel closely together like in a convoy. It requires, on the one hand, that each vehicle in the platoon control its velocity and the relative distance to the vehicle in front of it for avoiding rear collision and, on the other hand, that vehicles coordinate for enabling other vehicles to dynamically join or leave the platoon. This research investigated the mediated approach for architecting a platooning of self-driving vehicles, with SosADL, a novel SoS Architecture Description Language (ADL) enhanced with broadcasting for the Internet-of-Vehicles (IoV). In particular, it demonstrates how architectural mediators expressed with broadcast constructs of SosADL for IoV supports platooning architecture descriptions through an excerpt of a real application for architecting platoons of Unmanned Ground Vehicles (UGVs). This novel approach is supported by an integrated toolset for SoS architects. For details see: [13].

3.1.3 Enhancing SosADL with support for architecting exogenous SoSs

Keywords: Exogenous Architecture, Architecture Description Language (ADL), Software-intensive Systems-of-Systems (SoS), SosADL, Internet-of-Vehicles (IoV).

Nowadays, the Internet-of-Things (IoT) enables the engineering of software-intensive SoS, which are opportunistically constructed for achieving specified missions in specific operational environments. In particular, in the subset of IoT where "things" are predominantly connected vehicles, the so-called Internet-of-Vehicles (IoV), the challenge is to exogenously coordinate different vehicles for performing together, through emergent behavior, traffic-related missions, especially platooning. In platooning, two or more vehicles are connected together in convoy using wireless connectivity and automated driving support. The corresponding challenge in the architectural design of SoSs on IoV is to conceive concepts and mechanisms for describing how an SoS architecture

is able to create, on the fly, and maintain emergent behaviors from elementary connected vehicles, where the actual vehicles are not known at design time. To address this challenge, this research investigated the principle of supervenience for describing architecture-driven emergent behavior following an exogenous approach. In particular, we conceived novel concepts and mechanisms based on the novel π-Calculus for SoS, to support the architectural description of self-organizing SoSs, upwardly causing the required SoS emergent behaviors at run time. Especially, we demonstrated how architectural mediators expressed with SosADL in exogenous SoS architectures support vehicle platooning through an excerpt of a real application on the IoV. For details see: [8].

3.1.4 Enhancing SosADL with support for coping with uncertainty in SoSs

Keywords: Uncertainty, Architecture Description Language (ADL), Software-intensive Systems-of-Systems (SoS), Concurrent Constraints, SosADL, Internet-of-Thongs (IoT).

A challenging issue in the architectural design of an SoS is how to cope with the uncertainty raised by the limited knowledge of the operational environment where the SoS will actually be deployed as well as the constituent systems which will concretely participate in the SoS at run-time. It is especially the case of SoSs being architected on the Internet-of-Things (IoT). Indeed, due to the open and dynamic nature of the IoT, on the one hand, at design-time, most often the SoS architects do not know which will be the concrete IoT systems that will become constituents of an SoS, these being predominantly identified at run-time; on the other hand, the correct architecture depends not only on the constituent IoT systems but also, largely, on the operational environment where the SoS will be positioned on the IoT. The consequent research question is thereby how to design and describe the SoS architecture in a way that is flexible enough to cope with these different uncertainties. To address this challenge, this research investigated the notion of uncertainty in SoS and enhanced SosADL to cope with uncertainty in the architecture modeling of SoSs. It considers the concepts and constructs that makes possible to describe SoS architectures which will operate in unpredictable environments on the IoT based on the SosADL support for dealing with partial knowledge, grounded on concurrent constraints. For details see: [14].

3.1.5 Further enhancing SosADL with Digital Twins for coping with uncertainty in SoSs

Keywords: Uncertainty, Architectural Design under Uncertainty, Architecture Description Language (ADL), Software-intensive Systems-of-Systems (SoS), Concurrent Constraints, SosADL, Internet-of-Things (IoT).

When architecting SoSs in the Internet-of-Things (IoT), architects face two sorts of uncertainties. First, they have only limited knowledge about the operational environment where the SoS will actually be deployed. Second, the constituent systems which will compose the SoS might not be known a priori (at design-time) or their availability

(at run-time) is affected by dynamic factors, due to the openness of the IoT. The consequent research question is thereby how to deal with uncertainty in the design of an SoS architecture on the IoT. To tackle this challenging issue, this research addressed the notion of uncertainty due to partial information in SoS and proposed enhancements to the SoS Architecture Description language (SosADL) for expressing SoS architectures in the IoT under uncertainty. The core SosADL was extended with concurrent constraints and the concept of digital twins coupling the physical and virtual worlds. This novel approach is supported by an integrated toolset, the SosADL Studio. Validation results demonstrated its effectiveness in an SoS architecture for platooning of self-driving vehicles. For details see: [15].

3.1.6 Enhancing the implementation of SosADL by bridging Ecore and Coq: SosADL Type-Checker with Proof-Carrying Code

Keywords: Ecore, Coq, Proof-Carrying Code, Model Transformation, SosADL.

This research has been carried out in the context of the implementation of SosADL in terms of an Eclipse-based software environment. While SosADL has formal definition rooted in a π-Calculus enhanced with Concurrent Constraints, the π-Calculus for SoS, we have adopted the Eclipse ecosystem, including EMF, Ecore and Xtext for the convenience they provide in implementation tasks. Proof-carrying code is a well-known approach to ensure such an implementation involving non-formal technologies conforms to its formal definition, by making the implementation generate proof in addition to usual output artifacts. In this research, we investigated for an infrastructure that eases the development of proof-carrying code for an Eclipse/EMF/Ecore/Xtext-based tool, the SosADL Studio, in relation with the Coq proof assistant. At the core of this approach, we combine an automatic transformation of a metamodel into a set of inductive types, in conjunction with a second transformation of model elements into terms. The first one provides necessary abstract syntax definitions such that the formal definition of the language can be mechanized using Coq. The second transformation is part of the proof generator. For details see: [3, 4].

3.2 Supporting methods for architecting secure 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 SoSs, the focus of cybersecurity has been increasingly shifted to design and more recently to Cybersecurity by Design or Design for cybersecurity, where the SoS architecture is the keystone for enforcing cybersecurity.

In the sequel, the main results of this line of work produced in 2019 are presented.

3.2.1 Designing SoS architectures from mission definition

Keywords: Mission, Model-based Architecture Process, Systems-of-Systems.

SoSs require recurrent adaptation at runtime owing to the uncertainty and variability of the operational environment. Thus, during their execution, SoS can deviate from the initial specification, which is often a consequence of successive evolutions. This issue occurs mainly due to: (i) weak communication between the SoS analysis stage and architecture stage and (ii) the lack of links between the operational planning in the SoS analysis stage and systems that must be involved in the SoS architecture stage. This research proposed a model-based process that strengthens the links between the SoS analysis stage and the architecture stage in the wave life cycle. It ensures that the mission and role concepts for the SoS definition are sufficiently abstract to allow adaptation to the variability of the operational environment. This definition is translated into an abstract architecture that guides the choices of the system architect during the design and evolution stages. The supported language is an adaptation of the Systems Modeling Language (SysML) which can be further refined in terms of SosADL. Furthermore, we define a crowd management SoS to illustrate the process. For details see: [12].

3.2.2 Developing secure SoSs for rapid deployment

Keywords: Cybersecurity, Security Model at Architecture Design, Causal Chain, Systems-of-Systems..

In certain domains, such as secure humanitarian corridors in a conflict zone, a special type of SoS, needing a rapid deployment, has to be developed. Because of the tense time constraint, usually only a domain expert is responsible with this development. However, many such SoSs also have to take into account the security aspect. In this research we address the issue of "how to help a domain expert to integrate the security aspect into the rapid development of an SoS". In this work we developed an approach and a tool suite that help the domain expert tag business assets using security properties, which are then used to identify vulnerabilities and to propose possible security control mechanisms. For details see: [12].

3.2.3 Security analysis of SoSs in the IoT using attack trees

Keywords: Cybersecurity Analysis, Attack Trees, Rare Events, Internet-of-Things (IoT), Systems-of-Systems (SoS).

Attack trees are graphical representations of the different scenarios that can lead to a security failure. In combination with model checking, attack trees are useful to quantitatively analyse the security of a system or system-of-systems. Such analysis can help in the design phase of a system or SoS to decide how and where to modify the system in order to meet some security specifications. In this research we developed a security-based framework for modeling IoT systems or SoSs where attack trees are defined alongside the model. A malicious entity uses the attack tree to exploit the vul-

nerabilities of the system. Successful attacks can be rare events in the the system or SoS execution, in which case they are hard to detect with usual model checking techniques. Hence, in this research we use importance splitting as a statistical model checking technique for rare events. This technique requires a decomposition of an attack into sub parts, similarly to an attack tree. We argue that therefore, importance splitting is well suited, and benefits, from our modeling framework. We implemented this approach in a toolset and verified its effectiveness by running a set of experiments over a real-word example. For details see: [9].

3.2.4 Developing SoSs for the the battlefield: a proof-of-concept to deal with digitalization

Keywords: Military SoS, Battlefield Engineering, Model-Based Engineering, Operation Orders, Systems-of-Systems.

Digitalization of the whole society will change the way SoSs have to be considered. Remaining independently operated and managed, SoSs increase their collaboration skills using shared or cooperated information systems. People can be seen as particular digital subsystems due to smart equipments they can use. Military operations, which are considered as typical SoS, are no exception to this fact. New operational doctrines have to be created to take advantage of those new capabilities. In this research we developed new methods supported by tools inspired by Software Engineering to create new automated capabilities in battlefield engineering. They support the direction which should be considered in the area of battlefield engineering in order to deal with those new capabilities. Inspired from Model-Based Engineering, we realized a proof-of-concept showing how to change textual operation orders with graphical ones. The latter can be exported in a common standardized format, that enables digital interpretation. We present the OPORD-ML language which is based on a metamodel inspired from a NATO operation order standard. It is supported by an automatically generated tool. For details see: [10].

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 2019, the SosADL Studio includes the following modules.

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

Participants: Jérémy BuissonGersan Moguérou.

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. During 2019, work has been carried out to improve the error messages issued by the type-checker.

4.1.2 SosADL2Alloy: Generating concrete SoS architectures based on SosADL

Participants: Milena Guessi, Gersan Moguérou, 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 SAT solutions represent SosADL concrete architectures. During the integration of this module into the SosADL Studio, it has been improved to represent the generated concrete architectures in SosADL.

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 verified using ioSTS/Uppaal and 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.

This module has been integrated in the SosADL Studio,. Currently, this module translates SosADL concrete architectures into DEVSNL, the language supported by MS4ME. This dependence with MS4ME requires running the simulation on Windows or Linux.

4.1.4 SosADL2IoSTS: The SosADL support for architecture verification

Participants: Elena Leroux, Gersan Moguérou.

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

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

During 2019, all these modules and execution steps of the SosADL workflow were integrated into the SosADL Studio. We produced the SosADL eclipse plugin as an alpha version.

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 Engineering: Bilateral Collaboration on Cybersecurity in Software Architecture for Industrial Systems and Systems-of-Systems in the Industrial Internet-of-Things: Bilateral collaboration between ARCHWARE and the R&D division of SEGULA, a multinational system engineering company developing large-scale systems and systems-of-systems in different domains, including automotive, aeronautics, naval, and railway engineering. This ongoing collaboration aims to support the model-based engineering of critical systems-of-systems in the Industrial Internet of Things. The ARCHWARE team brings to this joint R&D project its expertise on formal approaches for the specification and verification of software architectures of systems-of-systems.

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)
- Jamal El Hachem has a collaboration on systems-of-systems security with Ali Babar (University of Adelaide) and another collaboration on securing systems-of-systems services with Aida CAUSEVIC (Malardalen University)

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:

- 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, 2019

  - MDE4IoT at MODELS: International Workshop on Model-Driven Engineering for the Internet-of-Things, 2019

- Isabelle Borne:
  - SESOS: ACM/IEEE ICSE International Workshop on Software Engineering for Systems-of-Systems, 2019

- Jérémy Buisson:
  - ICCS: International Conference on Computational Science, 2019
  - AICCSA: ACS/IEEE International Conference on Computer Systems and Applications, Conference Track on Software Engineering, 2019

- Régis Fleurquin
  - AICCSA: ACS/IEEE International Conference on Computer Systems and Applications, Conference Track on Software Engineering, 2019

- Flavio Oquendo:
  - ICSA: IEEE International Conference on Software Architecture, 2019
  - ECSA: European Conference on Software Architecture, 2019
  - SOSE: IEEE International Conference on System-of-Systems Engineering, 2019
  - SISOS: ACM Symposium On Applied Computing Program, Conference Track on Software Software-intensive Systems-of-Systems, 2019 (Chair)
  - ICT: International Conference on Telecommunications, 2019
  - CYBER: International Conference on Cyber-Technologies and Cyber-Systems, 2019
  - CCSE: Conference on Computer Science and Engineering, 2019
  - CODIT: International Conference on Control, Decision and Information Technologies, 2019
  - CEISEE: China-Europe International Symposium on Software Engineering Education, 2019
  - CPSIOT: International Conference on Cyber Physical Systems and IoT, 2019
  - ICSOFT: International Conference on Software and Data Technologies, 2019
  - SOFTENG: International Conference on Advances and Trends in Software Engineering, 2019
  - ICSEA: International Conference on Software Engineering Advances, 2019
  - ICONS: International Conference on Systems, 2019
  - ICAS: International Conference on Autonomic and Autonomous Systems, 2019
  - WCCS: World Conference on Complex Systems, 2019
  - SESOS: ACM/IEEE ICSE International Workshop on Software Engineering for Systems-of-Systems, 2019 (Chair)

  - COMPLEXIS: International Conference on Complexity, 2019
  - GE: ACM/IEEE ICSE Workshop on Gender Equality in Software Engineering, 2019
  - WSA: ECSA Track on Women in Software Architecture, 2019
  - CAL: French Conference on Software Architecture, 2019
  - SBES: Brazilian Symposium on Software Engineering, 2019
- Salah Sadou:
  - SESOS: ACM/IEEE ICSE International Workshop on Software Engineering for Systems-of-Systems, 2019

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 CZSF (Czech Science Foundation) on Complex 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)

Doctoral dissertations and "Habilitation" theses

[1] D. Beaulaton, *Security Analysis of IoT Systems using Attack Trees*, Theses, Université de Bretagne Sud, December 2019, https://tel.archives-ouvertes.fr/tel-02880109.

Articles in referred journals and book chapters

[2] C. Araujo, E. Cavalcante, T. Batista, M. Oliveira, F. Oquendo, "A Research Landscape on Formal Verification of Software Architecture Descriptions", *IEEE Access 7*, 2019, p. 171752–171764, https://hal.archives-ouvertes.fr/hal-02570179.

[3] J. Buisson, S. Rehab, "Effective Bridging Between Ecore and Coq: Case of a Type-Checker with Proof-Carrying Code", *in : Modelling and Implementation of Complex Systems*, November 2019, p. 259–273, https://hal.archives-ouvertes.fr/hal-01945245.

[4] J. Buisson, S. Rehab, "Generation of Inductive Types from Ecore Metamodels", *in : Model-Driven Engineering and Software Development. MODELSWARD 2018.*, February 2019, p. 308–334, https://hal.archives-ouvertes.fr/hal-02021361.

[5] I. Cherfa, N. Belloir, S. Sadou, R. Fleurquin, D. Bennouar, "Systems of systems: From mission definition to architecture description", *Systems Engineering 22*, 6, September 2019, p. 437–454, https://hal.archives-ouvertes.fr/hal-02368698.

[6] L. Garcés, F. Oquendo, E. Y. Nakagawa, "Software mediators as first-class entities of systems-of-systems software architectures", *Journal of the Brazilian Computer Society 25*, 1, December 2019, https://hal.archives-ouvertes.fr/hal-02570183.

[7] P. Gomes, E. Cavalcante, T. Batista, C. Taconet, D. Conan, S. Chabridon, F. Delicato, P. Pires, "A semantic-based discovery service for the Internet of Things", *Journal of Internet Services and Applications 10*, 1, December 2019, https://hal.archives-ouvertes.fr/hal-02147177.

[8] F. Oquendo, "Architecting exogenous software-intensive systems-of-systems on the internet-of-vehicles with SosADL", *Systems Engineering 22*, 6, September 2019, p. 502–518, https://hal.archives-ouvertes.fr/hal-02570186.

Publications in Conferences and Workshops

[9] D. Beaulaton, N. B. Said, I. Cristescu, S. Sadou, "Security Analysis of IoT Systems Using Attack Trees", *in : Graphical Models for Security*, M. Albanese, R. Horne, C. W. Probst (editors), Springer International Publishing, p. 68–94, Cham, 2019.

[10] N. Belloir, J. Buisson, O. Bartheye, "Metamodeling NATO Operation Orders: a proof-of-concept to deal with digitalization of the battlefield", *in : 2019 14th Annual Conference System of Systems Engineering (SoSE)*, IEEE, p. 260–265, Anchorage, France, May 2019, https://hal.archives-ouvertes.fr/hal-02181689.

[11] M. L. Kerdoudi, T. Ziadi, C. Tibermacine, S. Sadou, "Recovering Software Architecture Product Lines", *in : 24th International Conference on Engineering of Complex Computer Systems (ICECCS)*, IEEE, p. 226–235, Nansha, Guangzhou, China, November 2019, https://hal.archives-ouvertes.fr/hal-02268371.

[12] N. Messe, N. Belloir, V. Chiprianov, I. Cherfa, R. Fleurquin, S. Sadou, "Development of Secure System of Systems Needing a Rapid Deployment", *in : 2019 14th Annual Conference System of Systems Engineering (SoSE)*, Anchorage, United States, May 2019, https://hal.archives-ouvertes.fr/hal-02368826.

[13] F. Oquendo, "Architecting Systems-of-Systems of Self-driving Cars for Platooning on the Internet-of-Vehicles with SosADL", *in : 2nd IFIP International Internet of Things Conference (IFIP-IoT 2019)*, *IFIP Advances in Information and Communication Technology 574, Springer*, Springer, p. 3–20, Tampa, United States, October 2019, https://hal.archives-ouvertes.fr/hal-02570221.

[14] F. Oquendo, "Coping with Uncertainty in Systems-of-Systems Architecture Modeling on the IoT with SosADL", *in : 14th International Conference on System-of-Systems Engineering (SoSE)*, IEEE, p. 131–136, Anchorage, United States, May 2019, https://hal.archives-ouvertes.fr/hal-02570222.

[15] F. Oquendo, "Dealing with Uncertainty in Software Architecture on the Internet-of-Things with Digital Twins", *in : 9th International Conference on Computational Science and Its Applications (ICCSA 2019), LNCS 11619, Springer*, *9th International Conference on Computational Science and Its Applications (ICCSA 2019)*, Springer, p. 770–786, Saint Petersburg, Russia, July 2019, https://hal.archives-ouvertes.fr/hal-02570205.