# EDUFYSoS: A Factory of Educational System of Systems Case Studies

Antonia Bertolino*, Guglielmo De Angelis†, Francesca Lonetti*, Vânia de Oliveira Neves‡, and Miguel Angel Olivero*
*CNR–ISTI, Pisa, Italy
Email: {antonia.bertolino,francesca.lonetti,miguelangel.olivero}@isti.cnr.it
†CNR–IASI, Roma, Italy
Email: guglielmo.deangelis@iasi.cnr.it
‡Universidade Federal Fluminense, Niterói, Brazil
Email: vania@ic.uff.br

***Abstract*—We propose a factory of educational System of Systems (SoS) case studies that can be used for evaluating SoS research results, in particular in SoS testing. The factory includes a first set of constituent systems that can collaborate within different SoS architectures to accomplish different missions. In the paper, we introduce three possible SoSs and outline their missions. For more detailed descriptions, diagrams and the source code, we refer to the online repository of EDUFYSoS. The factory is meant to provide an extensible playground, which we aim to grow to include more systems and other missions with the support of the community.**

***Index Terms*—System of Systems, Software Testing, Educational Environment, SoS Factory, Testbed, Case Study**

## I. Introduction

We assist today to the growing digitalization of every aspect of modern business and societal life [1]. The spread of Internet for global communication, followed by the advent of Cyber-Physical Systems or more in general of the "Smart-Anything-Everywhere", has radically changed the information technology paradigms and the processes by which software systems are engineered. As early envisioned by Maier [2], the combination of pre-existing independent systems into a more complex System of Systems (SoS) allows for achieving a global collaborative *mission* beyond the goals and capabilities of the individual constituent systems. In this way, innovative distributed systems can be rapidly developed, for example to face emergency situations, but not only: domains of applications where the SoS paradigm is becoming prevalent include [3], [4], among others, transport networks, smart grids, e-commerce, homeland security, health care, military.

As we overview in the next section, research in SoS has been very active in recent years. SoS engineering poses several challenges relative to the modeling and management of the collaboration among their constituent systems. Depending on the SoS architecture, these could not be ready available to cooperate and adhere to the SoS directives. More challenging though is SoS validation, due to the concomitance of growing complexity with the (possibly even implicit) involvement of several stakeholders. On top of this, having been investigating SoS testing for some time [5], [6], we face another, practical, difficulty: the lack of SoS case studies on which proposed approaches can be validated and compared against each other.

The importance of common case studies has long been recognized by the software testing community, who has developed and shared several repositories for evaluating testing research, e.g., SIR[1] or Defects4J[2]. To the best of our knowledge no such repository exists for SoS researchers: several papers discuss scenarios in which a SoS is required, or describe, even in detail, the architecture of a SoS. However, they do not make publicly available the actual implementation of the presented SoS, which is hardly needed to assess testing techniques.

This paper fulfills such a need. To evaluate our research results in SoS testing, we resolved to develop ourselves a SoS case study and share it with the community. In order to achieve a realistic system, we deliberately followed a process that could resemble as closely as possible the engineering of a SoS in real world, and in particular we decided to build a SoS out of a set of pre-existing constituent systems that we could retrieve from the Internet. Accordingly, we decided to engineer a SoS in support of education, because in this domain it is easier to find several open-source constituent systems. Clearly, we acknowledge that an educational SoS is very broad and complex, beyond the mere software elements we can make available here, and the participation of its many stakeholders (students, teachers, administrators, etc) should be considered as well for analysis.

As the SoS was being built, it happened that different SoS case studies literally "emerged" from the combination of the same set of constituent systems, depending on how we specified the SoS mission. In the end, instead of choosing just one SoS implementation, we decided to make available the set of constituent systems, the process we followed, and examples of actual SoS instantiations to form what we called a "factory" of SoSs in the educational domain, or EDUFYSoS. In this way researchers can use the same kit to build and test different SoS architectures. We used the term factory also to signify that the

[1]see at https://sir.csc.ncsu.edu/portal/
[2]see at http://defects4j.org

repository is extensible and open to host other case studies.

For lack of space, in this paper we only provide an overview of EDUFYSoS and of its current contents. However, in the online material we provide more detailed descriptions of all components, and guidelines for usage, as well as the design and code of two SoS case studies.

The paper is structured as follows: in Section II we summarize background and related work; in Section III we outline the constituent systems and three different SoSs case studies they could form; in Section IV we briefly describe the available artefacts; finally conclusions are drawn in Section V.

## II. Background and Related Work

Since its appearance in the mid 90s, the SoS paradigm has been characterized in diverse ways. In the literature, four SoS architectures have been defined, i.e., Directed, Collaborative, Acknowledged, and Virtual, according to how the constituent systems are organized to accomplish the SoS mission [7]. The four architectures are distinguished depending on: *(i)* the existence of a central system that orchestrates the mission achievement, and *(ii)* the existence of guidelines for the interaction among the constituents. Precisely, in a directed architecture, a central constituent system is responsible to manage how the work is executed. In this case, the other constituent systems participate as slaves. In the acknowledged architecture, there exists as well a central system that monitors the mission; however, the other systems retain self control, hence the need of interaction guidelines to guarantee mission achievement. A collaborative SoS is similar to acknowledged ones, with the difference that no central entity exists. A set of constituent systems working collaboratively needs well-defined interaction guidelines and responsibilities. Finally, in a virtual architecture no central entity or guidelines for interaction exist and the joint work emerges spontaneously. The constituent systems working under this composition are offering their services without awareness of the mission to which they participate.

The current research in the SoS area mainly focuses on general problems, challenges and architectural descriptions of SoSs, whereas little attention is devoted to the investigation of case studies implementing SoSs. The work in [3] presents several SoS case studies in a broad set of domains including health care, finance, commerce, culture, security, military, transportation. The authors provide high level structures of the SoSs and describe their purposes, constraints and stakeholders targeting a high level worldwide audience. However, they do not provide implementation details and resources that can be used for software testing activity, which is our goal. Our proposal complements the work in [3] by describing three SoS case studies in the educational domain and providing among the other software artefacts also the source code.

The work presented in this paper spans over two research directions:

*a) SoS Modeling:* Model-based approaches [8] represent a promising direction for the analysis and development of SoS. Models are used in the different stages of SoS development, from mission model by the application domain expert to the architecture model by the system architect during the design and evolution stages of a SoS. New architecture description languages, such as SosADL [9] or COMPASS [3], have been specifically conceived to formally describe architectural emergent behaviors of SoSs. They represent formal specification languages providing novel architectural concepts and language constructs able to support SoS modeling and analysis. mKAOS [10] is a pioneering language that supports the specification of missions and the definition of relationships between such missions and the other elements of the SoS. The usage of this model allows to illustrate the delegation of responsibilities for each constituent system and how the composition of these responsibilities supports the main mission's achievement. The aim of our proposal is not to develop a specific model based approach for SoS definition. Instead, we provide mKaos and UML models of educational SoSs developed using mKAOS for the definition of the SoS mission and UML for the definition of their behavior. The mKaos models depict the constituent systems and the motivation for their participation in the SoS, in other words, the reason of their participation. On the other hand, the UML Use Cases show how the functionalities are being used within the SoS. These UML models depict the SoS architecture and the involved stakeholders on each SoS described in this study.

*b) SoS in educational context:* In the educational ecosystem, a growing amount of distributed information systems is dynamically adopted to support administrative and learning activities. Several online learning platforms such as Google Classroom [11], Moodle[4] or MOOCs [5] allow teachers to create online classrooms, while learners can attend the class and receive assignments. These learning platforms leverage the collaboration and interactions with other services provided by other independent systems. The SoS paradigm emerges as a complete solution to provide learners and teachers with an easy, time-saving and well-organized access to the resources [12], as well as to allow users to dynamically add a new independent system in the educational ecosystem as one of its members. In the last years, several works [12], [13], [14] show an increasing interest in building educational SoSs. Specifically, the authors of [12] present an orchestration of System of Information Systems (SoIS) with the aim to help learners to overcome the difficulties of managing different learning ecosystems, collaboratively accessing and sharing their resources from different learning platforms. The authors of [13] present an acknowledged SoS that allows to integrate processes, people and embedded devices belonging to an Internet of Everything (IoE) ecosystem and that operate into educational environments. The proposed SoS leverages a service-oriented architecture that offers services as interfaces between integrated systems. Finally, the work in [14] proposes an educational SoIS that combines functionalities of different

[3] see at http://www.compass-research.eu
[4] see at https://github.com/moodle/
[5] see at https://www.mooc.org/

independent virtual learning environments developed with different technologies and supported by various organizations. However, all the proposed solutions provide their architectural and behavioral models of educational SoSs, but do not provide any implemented instance of such systems. In our work, we model three reference SoSs from the educational domain and try to overcome the lack of a fully implemented solution by providing a factory of SoSs and a concrete implementation of two specific SoSs to be used for research in the SoS testing domain.

## III. Case Studies in Educational Environment

In this section, we present three different SoS case studies in the educational domain that emerge from the combination of a same set of constituent systems. Following the process presented in [15], we performed a domain analysis by identifying the actors and the set of constituent systems involved into the referred domain. Then, we elaborated three different SoS case studies starting from three different emerging missions.

The actors resulting from the domain analysis, and their major needs are:

- Students: a student should be able to learn contents related to a course appropriately. He/she should be able to see the list of available courses, join a course, attend a lesson in the on-line learning system and submit a deliverable. He/she should be also able to see all activities assigned to him/her in his/her calendar.
- Administrative staff: people belonging to the administrative staff are in charge of providing the students with a list of courses. They allow students to register to a course and enable the registration of student's marks.
- Teachers: a teacher is responsible for: managing his/her courses, scheduling the assignments and the online classes, evaluating students' deliverables and assigning marks to them.

The constituent IT systems that have been identified are:

- Administrative Office System (AOS): this system should have the ability to manage information about courses and students at university. An example of a concrete system having this capability is the secretariat of the university.
- Cooperative Administrative Office System (C-AOS): this system could be a physical or online system that allows to manage information about courses and students belonging to an international graduate programme defined across multiple universities. An example of such a system could be the online administration of an inter-university master's degree.
- Learning Management System (LMS): this system should deliver online educational courses, trace the students activities and report on their results. Concrete instances of this system could be: Moodle, FullTeaching, Google Classroom.
- Calendar System (CS): this represents a time-management system providing the user with the ability to manage appointments, events and deadlines. Examples of concrete systems that have this capability are Google Calendar, Yahoo Calendar.

Due to space limitation reasons, the full versions of the presented models are not reported in this paper. For the complete set of EDUFYSoS artefacts, the reader can refer to the EDUFYSoS repository [6].

### A. "Educational" System of Systems

In this section, we model a basic SoS called "Educational" SoS. The goal of this SoS is to allow students to attend online courses and manage the classes and their assignments in an integrated way. Specifically, this SoS involves all the identified actors, and the following IT constituent systems: AOS, LMS, and CS.

Fig. 1 shows the mission diagram of the "Educational" SoS developed using mKaos [10]. In this diagram, the blue rectangles represent the missions, the yellow circles represent the refinements of these missions, and the orange diamonds represent the capabilities required by the abstract constituent systems to achieve the referred mission. The identified high-level mission is "Students can follow courses at university". Following [15], we refined this abstract mission into three sub-missions that are: i) "Provide the students with the ability to choose the list of courses in the University"; ii)"Provide the students with the ability to attend online courses"; iii) "Provide the students with the list of courses/competences acquired during the degree". These sub-missions identify three different capabilities in the SoS, each of which can be independently achieved by a specific constituent systems (i.e, AOS, LMS, and CS).

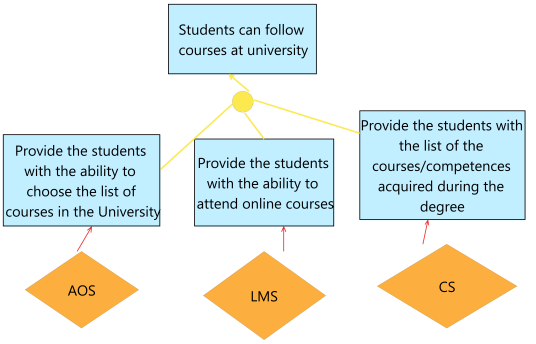


Fig. 1. "Educational" SoS mission diagram.

In Fig. 2, we present a UML use case diagram showing the interactions among the actors involved in the "Educational" SoS. In particular, the administrative staff can register a new course. The student can see the list of courses, enroll in a course, attends lessons, submit a deliverable and see his/her assignments directly in his/her calendar. The teacher can evaluate the students' deliverables and set the assignments for them.

"Educational" SoS can be either an acknowledged or a directed SoS depending on the instantiation of the constituent

[6]see at https://github.com/edufysos/edufysos

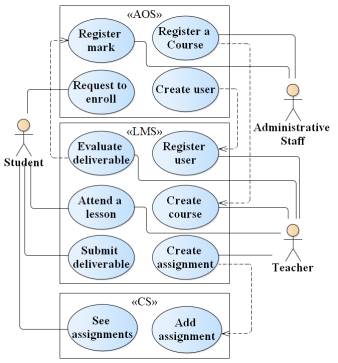


Fig. 2. "Educational" SoS use cases diagram.

systems and the existence of a centralised authority that coordinates their interactions. In particular, in "Educational" SoS, LMS is controlled by AOS. If CS is also controlled by AOS, the "Educational" SoS is a directed SoS, otherwise if the CS is managed by a different and independent organisation the "Educational" SoS is an acknowledged one. Specifically, in Section IV, we implement the "Educational" SoS as an acknowledged SoS. This "Educational" SoS will be also part of other two SoSs presented in Section III-B and Section III-C, respectively.

### *B. "International Master" System of Systems*

The "International Master" SoS refers to the case that a shared degree is offered by a set of universities with the aim of providing the student with a multicultural learning experience. The master is offered under the shared responsibility of the joining universities. In this sense, the administrative offices of each university manage the courses they offer within the international master program, also they cooperate in order to share the evaluations marked by the students during examinations. The considered SoS involves all the actors identified in Section III, and all the constituent systems as well.

Fig. 3 illustrates a use case that extends the previous use case given in Fig. 2. The universities cooperate by means of an interconnected network of AOS in order to proceed with the enrollment of the students. Several instances of AOSs are then coordinated by means of an instance of C-AOS that is responsible of delegating the obligations to each university. In this way, Students only interact with C-AOS and they do not need to enroll the same course in different AOS. Also, once enrolled, they can access the LMS from each participant university. The interactions among Student and both LMS and CS are the same foreseen by the use case in Fig. 2.

This SoS abides by an acknowledged architectural style. Indeed, the C-AOS is the one responsible for the accomplishment of the whole mission, while the AOSs belonging to different institutions maintain their technical and managerial independence.

### *C. "Student Mobility" System of Systems*

The "Student Mobility" SoS supports a mobility programme that allows students belonging to a university to make a study

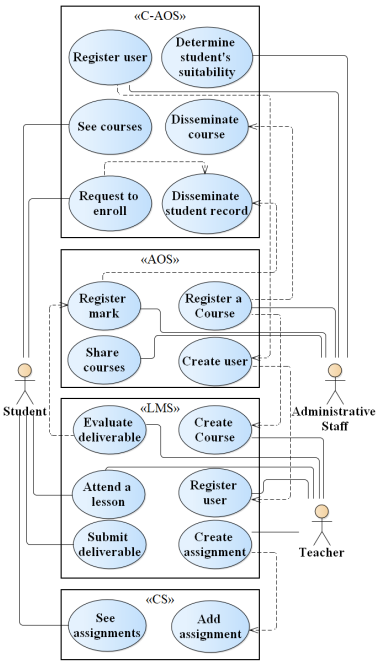


Fig. 3. "International Master" SoS use cases diagram.

period at a partner university (e.g., both in the same country or abroad like in the EU programme Erasmus). This exchange study period is part of the student's study programme to complete a degree. The SoS involves all the identified actors, and the following constituent systems: AOS, LMS, and CS.

This is a collaborative SoS that emerges from the collaboration of more universities, each one maintaining their operational, managerial and technical independence. A university voluntarily participates to the student mobility program; thus in a collaborative way the university accepts to fulfill a shared goal. In practice, this SoS can be seen as the composition of several "Educational" SoSs (as described in Section III-A) representing the different universities joining the mobility program. In particular, we distinguish the "origin" university as the university in which the student is enrolled, and the "destination" university as the partner university in which the student wants to perform an exchange period.

The high level mission of this SoS is refined into other abstract sub-missions, and is achieved by means of the collaboration between the "origin" and the "destination" universities. Fig. 4 shows an excerpt of the mission diagram of the "Student Mobility" SoS that illustrates how the high level mission "Provide the students with the ability to register in the selected courses in the partner university" is reached. The "origin" AOS provides the capability of requesting a student registration and the "destination" AOS grants the capability of registering the student to a course in the partner university.

The behaviour of the AOS differs from the one in the "Educational" SoS. In detail, the "origin" AOS provides the student with the list of courses in the partner university. Also it is in charge of requesting to the "destination" university the

enrolment of the student into the selected course. The AOS at the "destination" university can accept or deny the enrolment of the student. In case the student is registered to a course within the exchange program, she/he interacts with the LMS of the "destination" university in order to attend lessons, submit the deliverable and see all assigned tasks in the CS. Among the other use cases, the "destination" AOS is also in charge of notifying the students' marks on the AOS of the "origin" university. The behaviours of the LMS and CS do not differ from that in the "Educational" SoS of Section III-A.

The implementation of the mobility SoS is included in the EDUFYSoS repository and described in Section IV.

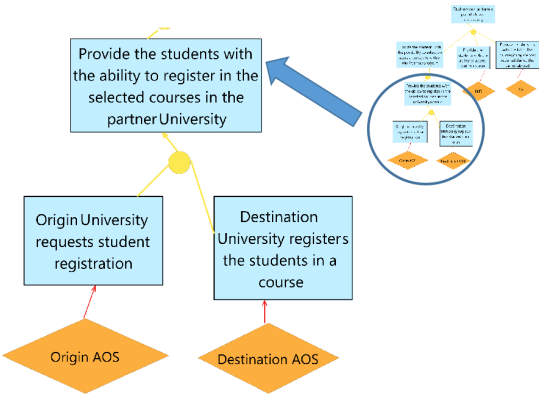


Fig. 4. An excerpt of "Student Mobility" SoS mission diagram.

## IV. EDUFYSoS

EDUFYSoS is offered as a means to create a family of SoSs related to an educational domain. To this purpose we provide: a set of educational SoSs case studies (introduced in Section III); the set of constituent systems that we can use to implement them; the process we followed to build two of these three case studies; and a set of useful guidelines supporting the instantiation of these case studies. Figure 5 presents an overview of EDUFYSoS.

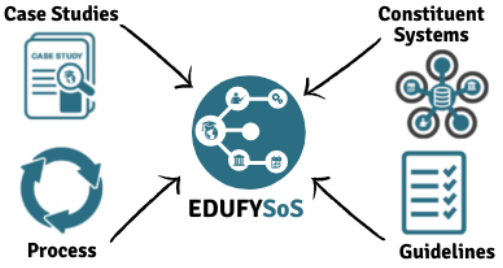


Fig. 5. EDUFYSoS overview.

Precisely, the artifacts produced, a copy of the open-source constituent systems, and detailed instructions for integrating these systems so to fulfill the designed missions can be found at the EDUFYSoS GitHub repository, available at https://github.com/edufysos. This section reports briefly about a blueprint for three out of the four constituent systems introduced in Section III: AOS, LMS, and CS. Instances of these IT systems can be deployed in order to set-up a SoS testbed for experimenting the evolution/variations of the case studies described in Section III.

In detail, to instantiate the SoS constituents EDUFYSoS refers to the following existing systems: AOS is implemented by means of `RosarioSiS`[7] which is an open-source project promoting a management information system for educational institutions. The `RosarioSiS` in the EDUFYSoS has been extended with specific plug-ins enabling both the students and the administrative staff to act according to the use cases introduced in Section III. The plug-ins are also responsible for managing the cooperation with other instances of constituents of either the same (i.e., AOS) or different type (i.e., LMS, and CS). Researchers aiming at building a testbed on EDUFYSoS can download and deploy one or more instances of the modified version of `RosarioSiS`. Each instance can be configured so to connect with its constituents and also with the AOS instances from other organisations in the SoS.

The reference implementation of LMS in EDUFYSoS is `Moodle`, for which no modifications have been put in place. To set-up an EDUFYSoS testbed, researchers have to deploy one or more instances of `Moodle` per organisation and configure each one so to bound instances of both AOS and CS. For example, the integration between instances of AOS, implemented by means of `RosarioSiS`, and instances of LMS, implemented by means of `Moodle`, is reported within the official documentation of the `RosarioSiS` Project [16].

As a further constituent in EDUFYSoS, the Google Calendar service has been identified as a reference implementation for the CS. Researches experimenting SoSs by means of EDUFYSoS are requested to federate instances of AOS and LMS with some Google Account on Calendar in order to be able to properly rely on its features. For example, configurations for a dynamic integration between `Moodle` instances and Google Calendar are available from the official documentation of `Moodle` [17].

The deployment of complex scenarios based on the considered constituent systems grants for the activation in the testbed of the IT procedures relative to the use cases presented in Section III. Among the others, an example of intra-organisational cooperation of the IT systems takes place when a new course will be created in an instance of the AOS: that course will be also created in the referred instances of the LMS. In addition, every time a teacher creates some assignments in the LMS, the students following the course will be able to see them in their own CS instances. These automatic interactions implement the behaviours foreseen in Section III-A.

As an example of inter-organisational cooperation among the IT systems, there is the "Student Mobility" SoS described in Section III-C. In this context, the instance of the AOS at the origin University is able to communicate with the respective AOS at the destination University. Specifically, the two IT systems (i.e., "origin" AOS, and "destination" AOS) are able to automatically enrol a student to perform his/her exchange

[7]see at https://github.com/francoisjacquet/rosariosis

period. The instances of AOS at both universities collaborate to enable the student to attend classes and to access the LMS of the destination University. When the student returns to his/her origin University, the "origin" and "destination" AOS automatically interact in order to report the credits for the attended courses so that they appear on his/her on-line curriculum.

## V. Conclusions

In this paper, we presented EDUFYSoS, a factory of educational SoS case studies to be used for research purposes in testing domain. Starting from an analysis of the educational domain, we modelled three different SoSs accomplishing different missions. We showed how different types of SoS emerged by the composition of the same set of constituent systems and how a SoS itself could be part of larger and more complex SoS. We provided the online repository of EDUFYSoS as a reference factory for the considered SoS. Indeed, EDUFYSoS also includes software resources that can be used to instantiate and set-up a SoS testbed implementing the modelled behaviours. All the artefacts (i.e., models, and software) could also be used for orchestrating evolution and modifications of the presented SoS.

Researches working with EDUFYSoS can experiment on *emergent behaviours* in SoSs by evolving the existing interactive use cases or by adding new ones. These modifications could focus on accessing technical APIs currently unused, or on consuming the technical interactions among constituents in a different way. For example, the implementation of emergent use cases foreseen for the C-AOS can be implemented in `RosarioSiS` by modifying the dedicated plug-ins that have been already developed or by providing new ones.

Independent upgrades in each reference implementation of a constituent opens to the possibility of experimenting approaches for governing/controlling the *opportunistic evolution* of a deployed instance of EDUFYSoS. Indeed, the lifecycles of the three constituents (i.e. `RosarioSiS`, `Google Calendar`, and `Moodle`) are independently managed; new features released in one of them could lead to the possibility to modify the way interactions take place in the baseline from EDUFYSoS. For example, an evolution in `Moodle` could lead to change how `RosarioSiS` enables course instances in the referred implementations of LMS. In the case of `Google Calendar`, opportunistic reactions to an emergent behaviour could come from the evolution of its public APIs.

In conclusion, EDUFYSoS promotes a blueprint for setting up testbeds of SoSs as it offers a baseline for enabling complex systems interactions in the educational domain. The deployment, and the interconnection of several instances of each constituent can be adopted by researchers in order to prepare replicable case studies for evaluating and comparing the proposed approach on SoSs.

In the future, we plan to extend the proposed factory with new constituent systems, model different missions as well as implement the case study presented in Section III-B. Future work on testing SoSs will leverage the current version of the factory and will likely enact further contributions for its evolution. For example, EDUFYSoS could be used as reference testbed for the evaluation of SoS testing approaches existing in literature, like [5], [6], and [18].

## VI. Acknowledgements

This paper has been partially supported by the Italian MIUR PRIN 2017 Project: SISMA (Contract 201752ENYB).

## References

[1] C. Legner, T. Eymann, T. Hess, C. Matt, T. Böhmann, P. Drews, A. Maedche, N. Urbach, and F. Ahlemann, "Digitalization: Opportunity and challenge for the business and information systems engineering community," *Business & Information Systems Engineering*, vol. 59, pp. 301–308, 07 2017.

[2] M. W. Maier, "Architecting principles for systems-of-systems," *Systems Engineering*, vol. 1, no. 4, pp. 267–284, 1998.

[3] A. Gorod, B. E. White, V. Ireland, S. J. Gandhi, and B. Sauser, *Case studies in system of systems, enterprise systems, and complex systems engineering*. CRC Press, 2014.

[4] C. B. Nielsen, P. G. Larsen, J. Fitzgerald, J. Woodcock, and J. Peleska, "Systems of systems engineering: basic concepts, model-based techniques, and research directions," *ACM Computing Surveys*, vol. 48, no. 2, pp. 1–41, 2015.

[5] V. O. Neves, A. Bertolino, G. De Angelis, and L. Garcés, "Do we need new strategies for testing systems-of-systems?" in *Proc. Int. Workshop on Software Engineering for Systems of Syestems*, 2018, p. 29–32.

[6] A. Bertolino, G. De Angelis, and F. Lonetti, "Governing regression testing in systems of systems," in *Proc. of Int. Symposium on Software Reliability Engineering Workshops*. IEEE, 2019, pp. 144–148.

[7] J. S. Dahmann and K. J. Baldwin, "Understanding the current state of US defense systems of systems and the implications for systems engineering," in *Systems Conference*. IEEE, 2008, pp. 1–7.

[8] I. Cherfa, N. Belloir, S. Sadou, R. Fleurquin, and D. Bennouar, "Systems of systems: From mission definition to architecture description," *Systems Engineering*, vol. 22, no. 6, pp. 437–454, 2019.

[9] F. Oquendo, "Formally describing the software architecture of systems-of-systems with SosADL," in *11th System of Systems Engineering Conference (SoSE)*. IEEE, 2016, pp. 1–6.

[10] E. Silva, T. Batista, and F. Oquendo, "A mission-oriented approach for designing system-of-systems," in *Proc. of System of Systems Engineering Conference*, 2015, pp. 346–351.

[11] S. Iftakhar, "Google classroom: what works and how," *Journal of Education and Social Sciences*, vol. 3, no. 1, pp. 12–18, 2016.

[12] M. Saleh and M.-H. Abel, "System of information systems to support learners (a case study at the university of technology of compiègne)," *Behaviour & Information Technology*, vol. 37, no. 10-11, pp. 1097–1110, 2018.

[13] R. A. Silva and R. T. Braga, "An acknowledged system of systems for educational internet of everything ecosystems," in *Companion Proc. of the European Conference on Software Architecture*, 2018, pp. 1–7.

[14] V. O. Neves, L. Garcés, and V. V. Graciano Neto, "Towards educational systems-of-information systems," in *Accepted for publication at Brazilian Symposium on Information Systems (SBSI)*, May 18–22, São Bernardo do Campo, Brazil, 2020.

[15] L. Garcés and E. Y. Nakagawa, "A process to establish, model and validate missions of systems-of-systems in reference architectures," in *Proc. of the Symposium on Applied Computing*, 2017, pp. 1765–1772.

[16] *Moodle integrator setup*, The RosarioSIS Project, – Official Documentation, accessed on 2020-02-24. [Online]. Available: https://gitlab.com/francoisjacquet/rosariosis/-/wikis/Moodle-integrator-setup

[17] *Using Calendar*, The Moodle Project, – Official Documentation, accessed on 2020-02-24. [Online]. Available: https://docs.moodle.org/35/en/Using_Calendar

[18] F. Zapata, A. Akundi, R. Pineda, and E. Smith, "Basis path analysis for testing complex system of systems," *Procedia Computer Science*, vol. 20, pp. 256–261, 2013.