REAP-SOS: A REQUIREMENT ENGINEERING APPROACH FOR SYSTEM OF SYSTEMS

Felipe Lima Duarte[1] and Angélica Félix de Castro[2] and Paulo Gabriel Gadelha Queiroz[3]

[1]Center of Exact and Natural Science, UFERSA, Mossoró - RN, Brazil

ABSTRACT

A System of Systems (SoS) is a class of system composed of a set or arrangement of independent systems that together provide unique functionality for the end user. Due to its complexity, the Requirement Engineering (RE) process needs to undergo adaptations to fit the development of this type of system. In this context, the objective of this work is to propose a approach for the development of SoS, called REAP-SoS. The main characteristic of the REAP-SoS is the derivation of the individual missions and requirements of the constituent systems based on the general SoS assignments. In addition, the approach is also able to derive the requirements of the constituent systems of SoS. To validate the approach, a case study on a SoS urban traffic control and monitoring was performed.

KEYWORDS

System of Systems, Requirements Engineering, SysML

1. INTRODUCTION

A SoS is defined as the result of a set or arrangement of independent systems that are integrated and combined. The result of this union provides unique capabilities to users [13]. To distinguish SoS from complex and traditional systems, it is necessary to understand some of its characteristics. According to Maier [9], a SoS presents five main characteristics: operational independence, managerial independence, evolutionary development, emergent behaviour and geographical distribution.

In a software development process, one of the main objectives is to define the functionalities and constraints of the system [15]. These definitions are made in the Requirements Engineering (ER) process. According to Adu [1], it is responsible for providing a suitable mechanism to understand what clients want, to analyze their needs, the feasibility of what is requested, to specify, to validate and to manage the requirements.

A Systematic Review (SR) was carried out in 2016 with the purpose of identifying specific RE approaches applied to SoS, and twenty five approaches have been found. Some of them try to cover the entire development process, and others focusing on specific steps, such as specification or requirements management. Another aspect about the approaches founded is that some aim to identify the requirements of the SoS as a whole and thus to select existing systems that meet these requirements, while others give greater importance in modelling the systems for a possible implementation.

In addition, as a result of SR, we found few studies with complete and well-documented methodologies, mainly regarding the design of the requirements of SoS and its Constituent Systems (CS). Many studies were limited to the SoS assignments or the requirements management activity.

Regarding traditional RE, existing approaches find it difficult to work in the context of complex systems, more specifically SoS, because of the requirements nature, which are fragmented, conflicting, unstable and often cannot be completely defined. Another issue is that, in SoS, there are two types of objectives, the objectives of the SoS as a whole; and also the objectives of individual systems [12].

Thus, the main purpose of this work is to present a specific approach to requirements engineering applied to SoS in such a way that it can guide the process of designing and modelling requirements. In addition, in order to validate the approach, we present a case study on a SoS for controlling and monitoring the urban traffic.

This work is organized as follows: section 2 presents related works; in section 3 presents the proposed approach; section 4 presents the case study and, finally, section 5 presents the final considerations of this work.

2. RELATED WORK

Some studies have already proposed specific RE approaches to the development of SoS, some of them are presented and discussed below.

The approach proposed by Holt et al. [7], is specific to SoS and was based on the traditional Context-Based Requirements Engineering (ACRE) approach, it contains three elements, which are:

- Ontology: used to describe concepts and terminologies of the application domain to be developed;
- Framework: defines a predefined set of system views, in practice, they are the artefacts to be created by the approach;
- Process: represented by a set of steps that are responsible for, on the basis of the ontology, generate the visions defined in the framework.

The approach proposes several steps in the process, among them, we can highlights: elicitation and development of requirements, verification and validation of requirements, control of requirements, traceability of requirements, monitoring of requirements, among others. We realized that this approach clearly represents what should be done to generate and manage the requirements, but does not offer many guidelines on how to do it.

Petrinca et al. [14] defines its approach as an iterative and top-down process, which begins with meeting stakeholder needs, represented by SoS missions. From these missions, a series of transformations in SysML models are carried out in order to derive the requirements of the Constituent Systems (CS). The process as a whole has been divided into three phases: definition of system context, definition of system behaviour and definition of architecture. It is an excellent approach for deriving the SOS missions and the requirements of the constituent systems, because it presents all the steps and transformations that must be made in the models generated by the approach. However, it does not address issues such as requirements management and validation. Another approach that deserves emphasis is defined by Lewis et al. [8], which propose a top-down approach, capable of understanding the requirements of the SoS as a whole, and bottom-up, capable of understanding the specificities of CS. It has the following phases: identifying the SoS context, identifying SoS and constituent objectives, understanding SoS interactions, identifying the capabilities and constraints of individual systems, and analyzing the gaps left by the process. It is a good approach to understanding the problem of defining and representing requirements in the context of SoS, but also does not offer many guidelines on how to do the steps described.

Other works that also address the theme of RE and SoS are: Ceccarelli et al. [5] and Yang-Turne et al. [21], which focuses mainly on the design and identification of requirements. Cavalcante et al. [4], which addresses the requirements modelling stage, proposing the use of implementations of the Goal Objective Requirements Engineering (GORE) approach, such as KAOS, i *, etc. Finally, the work of Vierhauser et al. [19] and Vierhauser et al. [20] addresses the requirements management stage, the latter proposing a monitoring approach based on three dimensions, namely: SoS requirements and SoS events.

From reading the works found in SR, which include the above works. With the exception of Petrinca et al. [14], the lack of a clear guide on how to generate the requirements of the constituent systems of SoS was perceived. Thus, the approach proposed in this work is intended to fill this gap left by other approaches. Regarding the approach proposed by Petrinca et al. [14], the main difference between the proposed approach and this work is the different activities of the process, such as modelling the structure, identify and model the missions of SoS, which we believe is easier to understand and use.

3. REAP-SOS: REQUIREMENTS ENGINEERING APPROACH FOR SOS

The REAP-SoS consists of a top-down proposal, which means that the requirements of CS are derived from the global SoS missions through iterations over SysML models. By missions, it is understood that they are typically viewed as goals, features, or a set of tasks. Missions can be related and contribute to the accomplishment of others, besides, as well as they can have a positive impact, they can also exert negative influences, making it impossible even to accomplish some task of the system. In SoS, there are two types of missions, which are: Individual Missions (IM), assigned to the constituent systems; and the Global Missions (GM), objectives assigned to SoS [17][18].

The goals of the proposed approach are:

- Analysis of SoS context and environment to be developed;
- Identify the CS of the SOS;
- Identification of the capabilities that are expected from the SoS as a whole, ie the general SoS;
- Transforming SoS missions into CS requirements;
- Modelling the SoS missions and CS requirements;
- Ensure the traceability of the requirements.

The approach, in addition to the identification of the SoS assignments and CS requirements, needs to address other aspects, such as: the type of SoS, the environment associated with SoS and the identification of interactions between systems. In addition to these, it is also important to understand the capabilities provided by individual systems.
The approach is composed of three elements, they are:

- Context Definition Phase: responsible for defining the entire context and environment to which SoS is embedded;

- Framework: responsible for defining the models and artefacts to be created by the approach;

- Conception and Modelling Phase: responsible for develop the elements of the framework.

With respect to traditional requirements engineering, the proposed approach differentiates itself by understanding from the beginning the complexity of the SoS assignments, thus trying to derive CS requirements through the global and individual SoS assignments. In addition, the process of identifying such systems, together with the implementation of a traceability study between requirements and constituent systems, are not characteristic of traditional approaches.

Figure 1 shows the REAP-SoS approach represented by the Business Process Model and Notation (BPMN) language. In the figure, we can see the iteration between the phases of context definition and conception and modelling phase. The latter is composed of some activities, which are: modelling the SoS structure, identifying and model the SoS missions, identifying, model and specify requirements, modelling system activities and modelling the system state. As can be seen from the BPMN loop notation, these activities are also performed iteratively and incrementally.

3.1. Context Definition Phase

The RAEP-SoS context definition is a phase responsible for identifying several aspects of the domain to which the SoS to be developed belongs. The first thing to be defined is whether the system to be specified really is a SoS. For this, the approach recommends identifying at least two CS that have the following characteristics: operational independence, managerial, emerging behaviour, evolutionary development and geographical distribution. In addition, such systems should contribute to a single specific purpose. The approach does not determine that all constituent systems should be identified, but it is important to try to find the majority of them because it will facilitate the next phases of the process.

The CS identification can be done by two main ways: by observing existing systems, trying to find some set of similar systems, analyzing their structure and their relationships. The other way is to apply questionnaires or perform interviews with experts on the problem that the SoS proposes to solve. In addition to these, Mokhtarpour et al. [10] propose an approach to CS selection for a SoS. Its methodology presents the following phases: identification of the missions, identification of candidate systems, selection of possible candidates, determination of alternatives and evolution of alternatives. Regardless of the manner used for identification. The approach recommends that for each system identified, a summary be made of it and describe its main objective in relation to SoS.

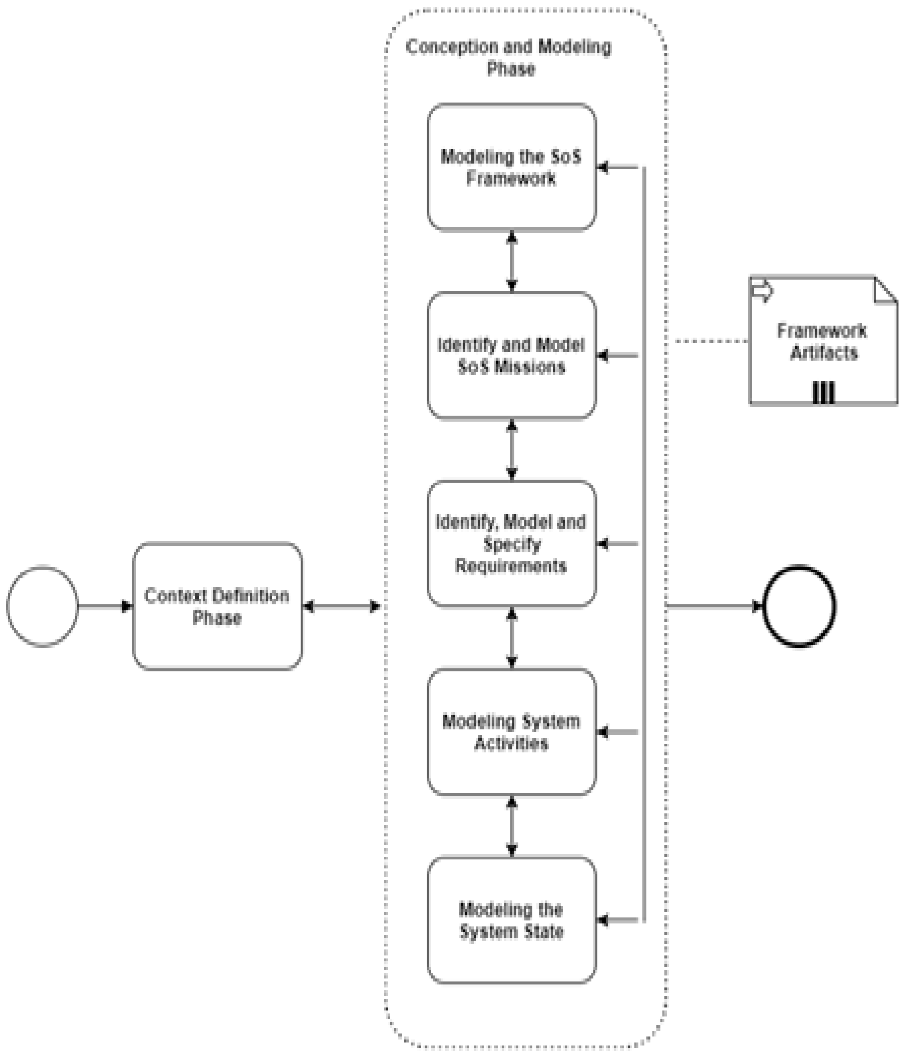

Conception and Modeling Phase
Modeling the SoS Framework
Identify and Model SoS Missions
Identify, Model and Specify Requirements
Modeling System Activities
Modeling the System State
Context Definition Phase
Framework Artifacts

Figure 1. REAP-SoS Approach in BPMN

After define if the system really is a SoS, the approach recommends identifying the environment to which the SoS is inserted, because each of the constituent systems of a SoS are influenced directly by its environment, the aspects of the environment to be defined are:

- Entity: is all that can influence SoS, for example: technologies, approaches, people, companies, stakeholders and others. It is important to emphasize that the concept of entity differs somewhat from the concept of stakeholders, since they do not necessarily have an interest in the project, but can only positively or negatively influence the SoS or the constituent systems;

- Influence: entities can and often exert different levels of influence between the constituent systems and SoS. A mapping of this influence is important to analyze and assist in prioritizing the changes in SoS and its systems.

Understanding these two aspects is important in anticipating future changes to the requirements of SoS as a whole and individual systems. The approach recommends the use of three types of influence, which are: low (L), medium (M) or high (H). The use of only three levels is due to the attempt of the approach not to generate excessive complexity. Another recommendation of the approach is to verify whether the entity can be seen as a stakeholder of a CS. If it is, it will have a greater degree of influence in this system and smaller in the other CS. If the entity cannot be seen as a stakeholder of any CS, it will probably have a greater degree of influence on the SoS as a whole.

To finalize the SoS context phase, the REAP-SoS recommends the creation of a feasibility study, to ensure that the SoS to be developed:

- Contributes to the organization's objectives;
- Can be built with the technology and staff available by the organization;
- Can be integrated with systems already present in the organization.

3.2. Framework

The framework is the element of the approach responsible for the formal definition of the artefacts that must be created by the conception and modelling phase, which are:

- Block definition diagram: responsible for modelling the SoS structure, presenting its CS;
- Block definition diagram (Missions): responsible for modelling the global and individual missions of SoS;
- Requirements diagram: responsible for representing the requirements of CS;
- Activity diagram: responsible for providing the dynamic structure of the systems, responsible for showing the flow of activities. showing how an activity depends on one another;
- State machine diagram: responsible for showing the possible states of an CS and the transactions responsible for its state changes.

3.3. Conception and Modelling Phase

The conception and modelling phase consists of a set of activities whose main purpose is to conceive and model CS requirements. It is important to note that they can and should be done in an iterative and incremental way. In addition, each activity is responsible for generating a framework artefact, as can be seen in Figure 2. The activities of this phase are: modelling the SoS structure, identifying and modelling SoS missions, identifying and modelling the requirements, model the static structure of the system and map the requirements. The main objective is to generate the artefacts required by the framework.

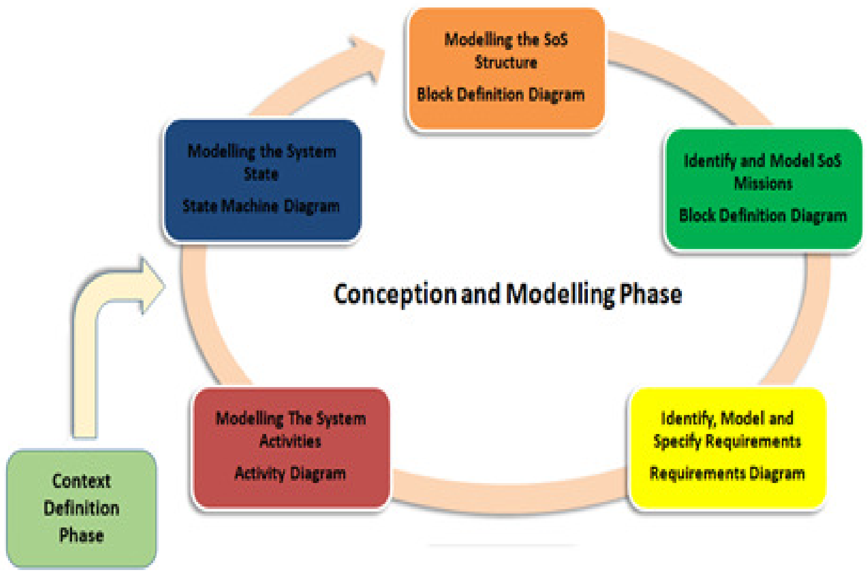

Modelling the SoS Structure
Block Definition Diagram
Identify and Model SoS Missions
Block Definition Diagram
Conception and Modelling Phase
Identify, Model and Specify Requirements
Requirements Diagram
Modelling The System Activities
Activity Diagram
Modelling the System State
State Machine Diagram
Context Definition Phase

Figure 2. Activities of the Conception and Modelling Phase

3.3.1 Modelling the SoS Structure

The purpose of this activity is to identify and formalize the SoS structure, identifying the CS which are part of SoS. It is also important to note that one SoS can be part of another SoS, this must also be modelled.

To model the general structure of SoS, the approach recommends that, given the constituent systems identified in the SoS context, model a SysML block definition diagram containing the main SoS systems identified so far and how they are related. It is important to make it clear that this is an early version of the diagram and that it can be changed after building other artefacts.

The approach also recommends inserting the <<SoS>> stereotype to identify the blocks as a system of systems, in addition to the stereotype <<CS>> to identify the block as a SoS constituent system.

The following is a summary of this activity, defining inputs, execution and output:

- Inputs: constituent systems identified in the context definition phase;
- Execution: create and model a block diagram to represent the SoS structure;
- Outputs: SoS structure (SoS and CS).

3.3.2 Identify and Model SoS Missions

The purpose of this activity is to identify and model the SoS and CS missions. For this, starting from the previous model, the blocks must be separated and for each of them their missions must be identified. If the block has the SoS stereotype, the missions identified will be global missions, if the block has the CS stereotype, the missions modelled will be individual missions of the system. The REAP-SoS approach then recommends techniques for identifying and modelling these missions, these techniques should be assisted by the entities defined in the context definition phase.

The REAP-SoS approach does not determine the techniques for identifying the SoS missions, instead it leaves the user free to choose the best form. However, the approach recommends two techniques, which are: personas and task analysis.

Due to the complexity of the system and the possibility of a large number of stakeholders from different domains, the approach recommends the use of a technique called personas. This technique assists the system users in understanding their characteristics, needs and objectives, thus supporting requirements engineering [3]. The personas technique mainly consists of collecting data about users, obtaining an understanding of their characteristics and defining descriptions for groups of users. Based on this understanding, focus on these personas throughout the software creation process [16]. The objective of using this technique is to have a real understanding of the different stakeholders of the various systems that make up the SoS, thus helping the requirements elicitation process.

Another technique is task analysis, it is a top-down approach, in which tasks of a higher abstraction degree are decomposed into sub-tasks and eventually detailed into events that describe it. The main goal of this approach is to build a hierarchy of tasks [22]. It can be observed that this concept fits well into the definition of the SoS missions, since the global missions are broken down into individual missions that carry them out, thus forming a hierarchy of missions.

To Model the SoS missions, the approach recommends the use of the SysML block definition diagram, since it is a general purpose diagram, there is no problem in using it more than once, in addition, since all other diagrams recommended by the REAP-SoS approach are made in SysML, it is interesting to model the missions also with this language. The difference here is that if the missions were derived from a block with stereotype <<SoS>>, then the missions must have the <<Global Mission>> stereotype, if they are derived from CS, they should have the <<Individual Mission>> stereotype. Besides this information the approach also recommends that the block (SoS or CS) that originated the mission be placed in stereotype form. It is observed that a tree structure will form and probably the leaves of the trees will be the individual missions of the constituent systems.

A summary of this activity is:

- Inputs: SoS structure (SysML block definition diagram) and entities of the context definition phase;

- Execution: for each block of the SoS structure, create and model their missions, recommended techniques: personas, task analysis;

- Outputs: SoS and CS missions (SysML block definition diagram).

3.3.3 Identify, Model and Specify Requirements

The objective of this activity is to identify the main functionalities of the systems that will be responsible for carrying out the SoS mission. These should have a high degree of abstraction, characterizing them as CS user requirements. For this, starting from the previous model, for each identified individual mission, the requirements that fulfil this mission must be derived.

In order to carry out, as in the previous phase, stakeholders (entities) should be consulted. As it is CS, more traditional techniques are recommended, such as: interviews, questionnaires, ethnography and brainstorming. For requirements modelling, the approach recommends the use of SysML requirements diagram.

A summary of this activity is:

- Inputs: SoS and CS missions (SysML block definition diagram);
- Execution: for each mission, to create and model the requirements that fulfil them, recommended techniques: interviews, questionnaires, ethnography and brainstorming;
- Outputs: CS requirements (SysML requirements diagram).

3.3.4. Modelling the System Activities

The main objective is the specification of the behaviour of the SoS as a whole and how the collaboration of CS occurs, from the functional point of view. In addition, specify what will be subsequently designed, or directly constructed, thereby reducing the abstraction level of the scope, making it easier to understand what has to be done by the developers.

A summary of this activity is:

- Inputs: SoS structure (SysML block definition diagram) and CS requirements (SysML requirements diagram);
- Execution: create and model the activities of the systems that make up the SoS;
- Outputs: SoS and CS activities (SysML activity diagram).

3.3.5 Modelling the System State

The objective of this activity is to provide a better understanding of the CS and to facilitate the modelling of this system for the project team. This is done by creating state diagrams of the constituent systems.

A summary of this activity is:

- Inputs: SoS structure (SysML block definition diagram) and CS requirements (SysML requirements diagram);
- Execution: if the states of the systems can be easily identified, create and model its states;
- Outputs: CS states (SysML state diagram).

4. CASE STUDY

The present case study aims to present the use of the REAP-SoS approach. It consists of designing and modelling the requirements of a SoS for controlling and monitoring urban traffic.

Nowadays, heavy traffic and the increase in the number of traffic accidents have caused a high cost (economic, social and environmental) for companies. In Brazil, especially in large metropolises, public managers have one of their main challenges in urban mobility. Several alternatives have been and are made to solve this problem, among them, we can mention: improvement of infrastructure, traffic restriction program (To combat air pollution and traffic jam the city only allows vehicles whose licence numbers end with certain digits to drive on particular weekdays.) and collection of fees for urban areas. However, they did not have the desired effect [2].

The use of technologies in transport and vehicle infrastructure results in so-called Intelligent Transport Systems (ITS) [6]. These systems have, among other objectives, to provide efficient telecommunication and computer solutions for urban traffic problems. Among the objectives of these systems can be highlighted: traffic control and traffic lights, management of emergency services, automatic collection of tariffs in collective transportation, among others.

The context of ITS is directly related to a relevant theme, which is the "Smart Cities", which can be seen as a set of steps that a citizen takes, together with services and technologies to make the city a more habitable environment/comfortable, with more efficient services and ready to suit any situation. Among the essential aspects for the creation of a smart city, it is necessary a great modern and intelligent digital infrastructure able to meet the demands that these cities need [11].

4.1. Context of Urban Monitoring and Control

In this section, as determined by the REAP-SoS approach (Section 3.1), the CS, the SoS existence check, the SoS type definition, as well as the SC entities for the urban traffic monitoring and control system are defined and presented. All information, as well as the missions defined subsequently, were taken from the observation and reading of the work related to the SIMTUR project, which can be accessed at the following address: *http://projeto.unisinos.br/simtur*. The following are the CS identified:

- Smart Cars (CS1): this type of system, in addition to all the functionalities of a conventional car, would also be the primary agent responsible for collecting information about traffic and sending it to the traffic control system. Based on the information collected in real time, the decisions about the best routes and the following routes would also be taken in real time;

- Smart buses (CS2): many people rely on collective transportation systems to move around in big cities, so, in addition to smart cars, it is also very important to have intelligent public transport systems. These, in addition to all the functionalities of smart cars, with regard to improving urban traffic, would also be responsible for collecting and reporting data such as: route performed on the day, location, average time at each stop or terminal, as well as other information;

- Waze (CS3): it is known that cars and buses with the features described above are practically unviable nowadays, from an economic point of view. For this, a cheaper alternative would be the use of applications such as *Waze* (*https://www.waze.com*) or similar (*Google Maps*, *Tom Tom Go*). This application would be responsible for sending the vehicles location to the traffic control system;

- Smart traffic light (CS4): based on information collected in real time, the intelligent traffic lights would be the main actuators in the control flow of the main roads in the cities, having as main objective the control of the time of opening and closing based on the information of the most congested routes;

- Traffic controller (CS5): this system would be the main driver of urban traffic in order to reduce congestion. It would be responsible for receiving information from intelligent vehicles, identifying possible bottlenecks and triggering intelligent traffic lights;

- Public transportation application (CS6): this system is responsible for providing the user with information about bus stations, routes, which bus is closest, how long to reach the bus stop, among others.

After identifying the systems, it is necessary to verify if they have the characteristics of a SoS. The following is a brief summary of two of the five systems identified (smart cars and smart traffic lights).

In the case of smart cars, the main features that characterize it as a SoS constituent system are:

- Operational Independence: all cars can be seen as a system that operates without the need for other systems, for example: a car operates without the need for other vehicles to be running;

- Management Independence: all the functionalities present in cars, be they related to locomotion or information collection are controlled by the vehicle itself, without the interference of other systems;

- Geographic distribution: each vehicle occupies and operates in a space geographically distant from other systems;

- Emerging Behaviour: each car can collect and pass on information to both the central control units and other vehicles.

In the case of intelligent traffic lights, the main aspects that characterize it as a constituent system of SoS are:

- Operational Independence: a semaphore shall operate independently of the operation of other traffic lights or vehicles and systems;

- Management Independence: each traffic light can be managed independently of other traffic lights. Although, the information obtained by the control system will affect its management;

- Geographic distribution: each traffic light will be geographically separated by city roads;

- Emerging Behaviour: control the flow of cars, improving congestion control.

After identified the CS and ensuring that the system really is a SoS, the next step is to define the entities and the degree of influence they have on systems, the entities identified are

- Drivers: responsible for driving vehicles on the roads of the city;

- Users of public transport: users of collective transportation systems such as buses, subways etc;

- Pedestrians: although they do not have a direct influence on the problems that the SoS propose to solve, they are part of the organization chart of the traffic in the big cities;

- Technology: the entity that makes everything possible, since all the solutions go through the development of a great technological apparatus;

- Users of Traffic Control System (TCS): users of traffic control systems. Although this system can be autonomous, there must be users to manage and control some aspects of this system;

- Users of the Public Transport Control System (PTCS): as in the previous entity, there may also be users in the public transport control system;

- Local Government: probably the financier of the project, since it is the function and duty of public managers to solve the problems of mobility in their cities;

- Public transportation concessionaire company: in Brazil, most of the transportation companies are private and have partnerships with the government, and can exploit that activity over a period of time. Being part of the solution directly, it is also an entity interested in the SoS project.

Regarding the influence that these entities exert on the systems, in Table 1, a summary is presented. Following the approach, three degrees of influence were defined: low (L), medium (M) or high (H). Entities such as: drivers and users of public transport, in addition to the degree of influence in systems such as automobiles and public transport, also have a high degree of influence in the SoS as a whole, since they are the main stakeholders in SoS.

Table 1. Levels of Influence of Entities.

Entities	**CS1**	**CS2**	**CS3**	**CS4**	**CS5**	**SoS**
Drivers	H	L	M	L	L	H
Users of public transport	L	H	M	L	L	H
Pedestrians	L	L	H	L	L	L
Technology	H	H	H	H	H	H
Users of TCS	M	L	H	H	M	H
Users of PTCS	L	M	M	L	H	M
Local Government	M	M	H	H	H	H
Public transportation concessionaire	L	H	M	L	M	M

To conclude, the approach recommends conducting a feasibility study. It aims to assess whether the system can actually be built, analyzing aspects such as: available technology, time, personnel and cost.

As this case study is only intended to present and validate a concept, as well as to deepen the knowledge of designing and modelling the requirements of a SoS using REAP-SoS. The feasibility study was not created.

4.2. Modelling the SoS Structure

As recommended by the REAP-SoS approach (section 3.3.1), the SoS structure for urban traffic control and monitoring is presented, as can be seen in Figure 3. The stereotype <<SoS>> is placed to identify the blocks as a system of systems, and the stereotype <<CS>> is inserted to identify the block as a SoS constituent system.

The main SoS is the *Traffic Control and Monitoring System*, which consists of two other SoS, which are the *Congestion Control System* and *The Public Transport Information System*. For once, these SoS are composed of several CS, which are the *Traffic Controller*, the *Smart Traffic Light*, the *Smart Bus*, among others. It is also important to note that the *Information Capture System* (CS) can be instantiated either by *Waze* or *smart car* or *Smart Bus*, the latter, beside participate in the *Information Capture System*, also participates in the *Public Transport Information System*.

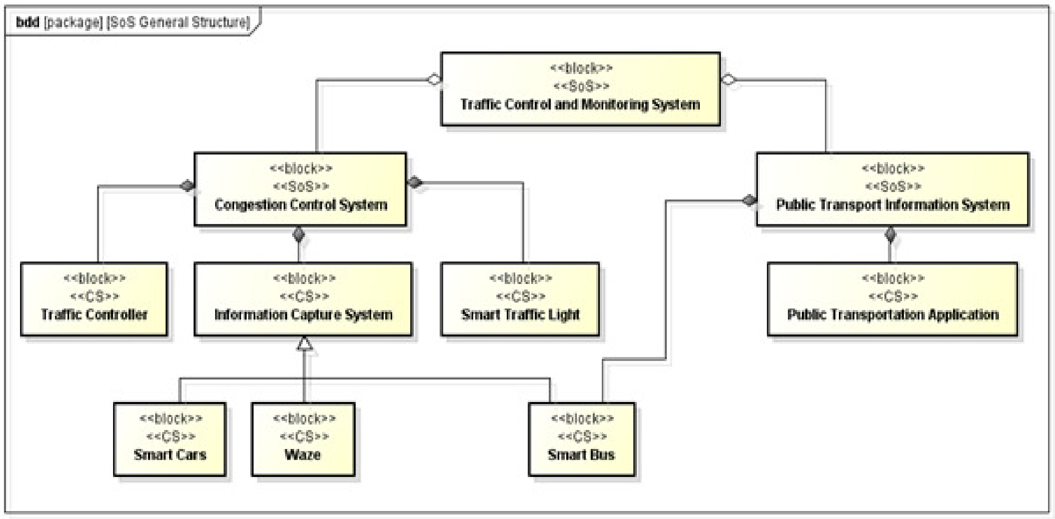
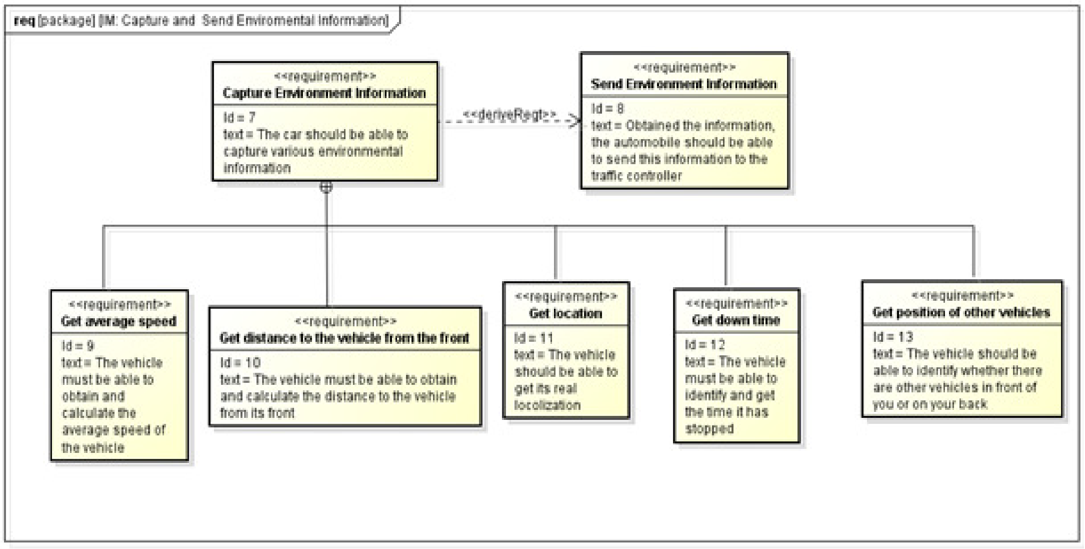

bdd [package] [SoS General Structure]
<<block>> <<SoS>> Traffic Control and Monitoring System
<<block>> <<SoS>> Congestion Control System
<<block>> <<SoS>> Public Transport Information System
<<block>> <<CS>> Traffic Controller
<<block>> <<CS>> Information Capture System
<<block>> <<CS>> Smart Traffic Light
<<block>> <<CS>> Public Transportation Application
<<block>> <<CS>> Smart Cars
<<block>> <<CS>> Waze
<<block>> <<CS>> Smart Bus

Figure 3. SoS general structure

4.3. Identify and Model SoS Missions

Following the REAP-SoS approach (section 3.3.2). The global and individual missions of SoS and CS have been defined and modelled. As can be seen in Figure 4, each block is represented by a mission, the stereotypes determine whether the mission is global or individual. In addition, a stereotype of the block (system) that originated the mission is added.


req [package] [IM: Capture and Send Enviromental Information]
<<requirement>> Capture Environment Information
<<deriveReqt>>
<<requirement>> Send Environment Information
<<requirement>> Get average speed
<<requirement>> Get distance to the vehicle from the front
<<requirement>> Get location
<<requirement>> Get down time
<<requirement>> Get position of other vehicles

Capture Environment Information: Id = 7, text = The car should be able to capture various environmental information

Send Environment Information: Id = 8, text = Obtained the information, the automobile should be able to send this information to the traffic controller

Get average speed: Id = 9, text = The vehicle must be able to obtain and calculate the average speed of the vehicle

Get distance to the vehicle from the front: Id = 10, text = The vehicle must be able to obtain and calculate the distance to the vehicle from its front

Get location: Id = 11, text = The vehicle should be able to get its real locolization

Get down time: Id = 12, text = The vehicle must be able to identify and get the time it has stopped

Get position of other vehicles: Id = 13, text = The vehicle should be able to identify whether there are other vehicles in front of you or on your back

Figure 4. SoS Missions

The global missions identified are: managing and monitoring urban traffic, controlling traffic congestion, and monitoring and reporting the status of public transportation. The individual missions identified are: identifying congestion locations, controlling vehicle flow, capturing and sending information from the environment, and finally, consulting itineraries and bus stops.

4.4. Identify and Model Requirements

As recommended by the REAP-SoS approach (section 3.3.3), after the modelling of the missions, the requirements diagram was created. As can be seen in Figure 5, the requirements that accomplish the individual mission *Capture and Send Environmental Information* are presented. In this model, two main requirements are presented, *Capture Environment Information* and *Send Environment Information*, the latter, to be performed accurately That the former is implemented, as can be seen in the notation <<*deriveRect*>>. The *Capture Environment Information* requirement consists of several other requirements, such as: *average speed*, *time stopped*, *identifying other vehicles* and so on. They are all responsible for capturing a type of environmental information, such as: average speed of the vehicle, identification of other vehicles on the road, vehicle stopped time etc.

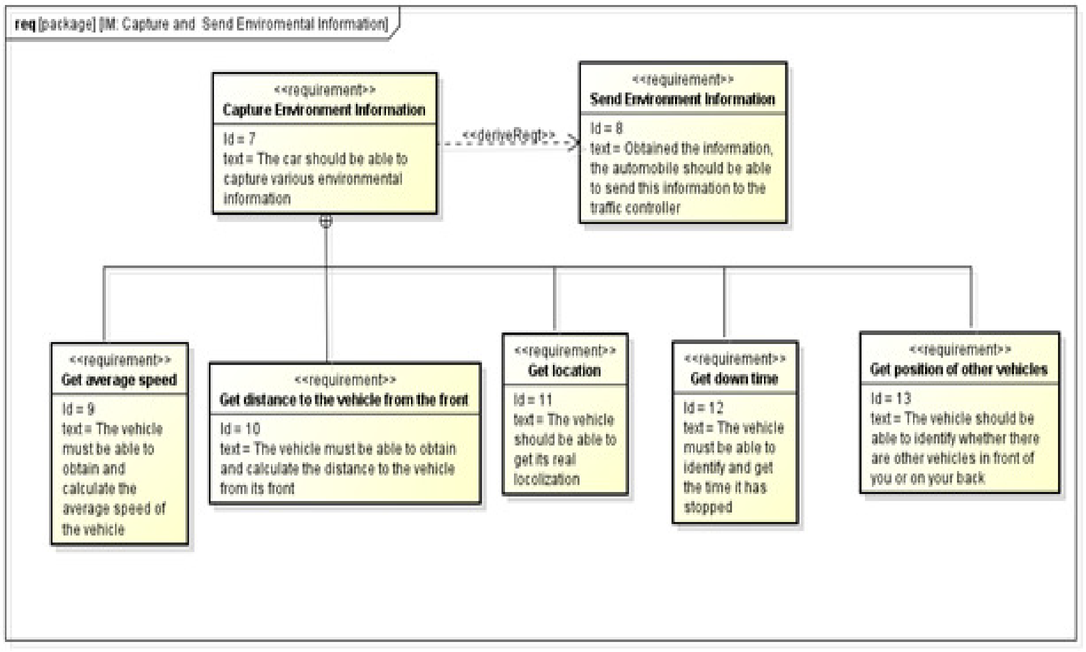

req [package] [IM: Capture and Send Enviromental Information]
<<requirement>>
Capture Environment Information
Id = 7
text = The car should be able to capture various environmental information
<<deriveReqt>>
<<requirement>>
Send Environment Information
Id = 8
text = Obtained the information, the automobile should be able to send this information to the traffic controller
<<requirement>>
Get average speed
Id = 9
text = The vehicle must be able to obtain and calculate the average speed of the vehicle
<<requirement>>
Get distance to the vehicle from the front
Id = 10
text = The vehicle must be able to obtain and calculate the distance to the vehicle from its front
<<requirement>>
Get location
Id = 11
text = The vehicle should be able to get its real locolization
<<requirement>>
Get down time
Id = 12
text = The vehicle must be able to identify and get the time it has stopped
<<requirement>>
Get position of other vehicles
Id = 13
text = The vehicle should be able to identify whether there are other vehicles in front of you or on your back

Figure 5. Smart Car Requirement Diagram

4.5. Modelling Activities of the Constituent Systems

As recommended by the REAP-SoS approach (section 3.3.4), an activity diagram was created for the *Congestion Control System*, as can be seen in Figure 6. The intention is to present the flow of information between the constituent systems of this SoS. *Smart cars*, *Smart buses* and *Waze* get the necessary information, send this information to *Traffic Controller*, which is responsible to processes, identifies congestion and sends this information to the *Traffic Lights*, which in case of congestion changes its configuration.

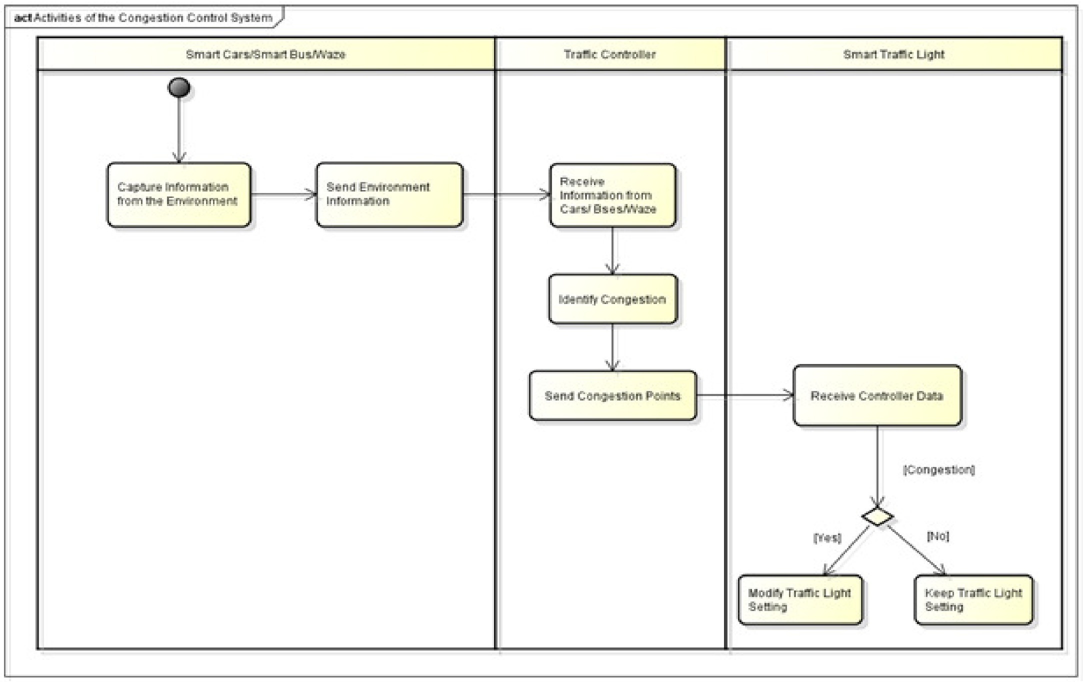

act Activities of the Congestion Control System
Smart Cars/Smart Bus/Waze
Traffic Controller
Smart Traffic Light
Capture Information from the Environment
Send Environment Information
Receive Information from Cars/ Bses/Waze
Identify Congestion
Send Congestion Points
Receive Controller Data
[Congestion]
[Yes]
[No]
Modify Traffic Light Setting
Keep Traffic Light Setting

Figure 6. Activities of Control Vehicle Congestion SoS

4.6. Modelling the System State

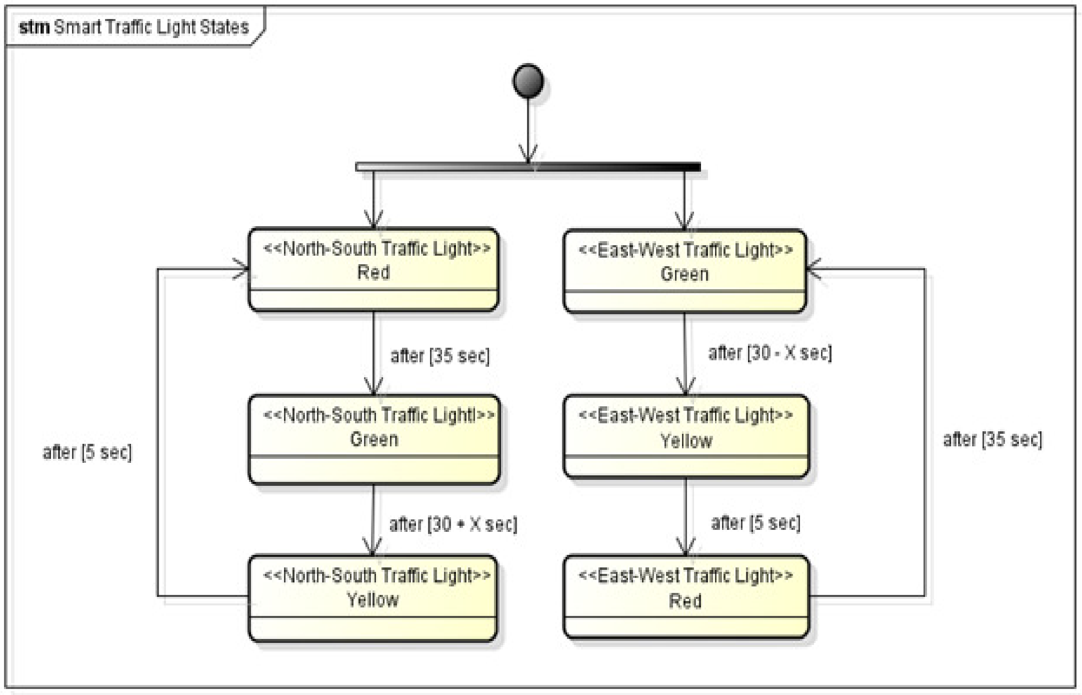

stm Smart Traffic Light States
<<North-South Traffic Light>> Red
<<East-West Traffic Light>> Green
after [35 sec]
after [30 - X sec]
<<North-South Traffic Lightl>> Green
<<East-West Traffic Light>> Yellow
after [5 sec]
after [35 sec]
after [30 + X sec]
after [5 sec]
<<North-South Traffic Light>> Yellow
<<East-West Traffic Light>> Red

Figure 7. States of Smart Traffic Light System

As discussed in section 3.3.5, the *Smart Traffic Lights* state diagram was created to show the operation of this system. As can be seen in Figure 7, there are two modelled traffic lights, which represent the street with north-south direction and the other that represents the street with east-west direction, the stereotype is identifying which semaphore the state belongs to. Three states were defined: *Green*, *Yellow* and *Red*. The main difference from this traffic light to the conventional one is the use of Variable X in the transition from green to yellow state (both signals), note that while it decreases the time in a traffic light, it increases the time in another, that variable will allow to modify the configuration of the traffic light and to provide a longer stay from the green to the more congested street.

5. CONCLUSIONS

This paper presented the REAP-SoS approach, which is an RE approach applied to the development of SoS. It is composed of three elements, they are: the context of the SoS, responsible for the definition of the environment to which the SoS is inserted; Design and modelling, responsible for describing the stages responsible for generating the SOS missions and the CS requirements; and finally; The framework, responsible for the formal definition of the artefacts that must be created by the process.

The proposed approach was based on other approaches specific to the development of SoS. Reap-SoS tries to unite the best of these methodologies, besides inserting and modifying elements in order to obtain a robust and complete approach, mainly in the design and modelling stage of the Requirements of SoS as a whole and CS.

In addition, in order to validate the approach, a case study on a SoS for control and monitoring of urban traffic was carried out. This system main purpose is to improve the flow of vehicles, reducing congestion and facilitating the life of the population of big cities. As it is a complex system with many CS, the use of a specific approach to SoS was necessary.

We believed the Reap-SoS can contribute to the subject of study, as well as serve as a basis for others approaches. In addition, it is expected to be the beginning of a comprehensive approach that will encompass all stages of the development of a SoS. Thus, as a starting point, it is expected that in the future, we also address architecture design, implementation, testing and management of SoS.

REFERENCES

[1] Adu, Michael (2014) “Inadequate Requirements Engineering Process: A Key Factor for Poor Software Development in Developing Nations: A Case Study”, International Journal of Computer, Electrical, Automation, Control and Information Engineering, vol 8.

[2] Buarque, S. C. (2008) “Construindo o desenvolvimento local sustentável”, Garamond.

[3] Castro, John W. & Acuña, Silvia T. & Juristo Juzgado, N. (2008) “Enriching requirements analysis with the personas technique”, First Workshop on the Interplay between Usability Evaluation and Software Development.

[4] Cavalcante E. & Batista T. & Bencomo N. & Sawyer P. (2015) “Revisiting Goal-Oriented Models for Self-Aware Systems-of-Systems”, IEEE International Conference on Autonomic Computing, 231—234.

[5] Ceccarelli A. & Mori M. & Lollini P. & Bondavalli A. (2015) “Introducing Meta-Requirements for Describing System of Systems”, IEEE 16th International Symposium on High Assurance Systems Engineering, 150—157.

[6] Colombo, A. & Del Vecchio, D. (2012) “Efficient Algorithms for Collision Avoidance at Intersections”, 15th ACM international conference on Hybrid Systems: Computation and Control, pp. 145—154.

[7] Holt J. & Perry S. & Payne R. & Bryans J. & Hallerstede S. & Hansen F. O. (2015) “A Model-Based Approach for Requirements Engineering for Systems of Systems”, IEEE Systems Journal, 252—262.

[8] Lewis G. A. & Morris E. & Place P. & Simanta S. & Smith D. B. (2009) “Requirements Engineering for Systems of Systems”, 3rd Annual IEEE Systems Conference, 247—252.

[9] Maier, M.W. (1996) “Architecting principles for systems-of-systems.” In INCOSE International Symposium 6 (1):,565-573.

[10] Mokhtarpour B. & Stracener J. (2014) “A Conceptual Methodology for Selecting the Preferred System of Systems”, IEEE Systems Journal, 1—7.

[11] Monzon, A. (2015) “Smart cities concept and challenges: Bases for the assessment of smart city projects”, Smart Cities, Green Technologies, and Intelligent Transport Systems, Springer, Cham, pp. 17—31.

[12] Ncube C, (2011) “On the Engineering of Systems of Systems: key challenges for the requirements engineering community”,{Workshop on Requirements Engineering for Systems, Services and Systems-of-Systems, 70—73.

[13] OUSD(AT&L), DoD. (2008) “Systems and Software Engineering. Systems Engineering Guide for Systems of Systems”. Technical Report Version 1.0. Office of the Deputy Under Secretary of Defense for Acquisition and Technology, Department of Defense.

[14] Petrinca P. & Gammaldi M. & Tirone L (2012) “A SysML-based Approach for the Specification of Complex Systems”, INCOSE International Symposium, 713—786.

[15] Ribeiro, L. C. M. & Ramos, C. S. & Brito, M. F. & Figueiredo, R. M. C. (2011). Definição de um processo de engenharia de requisitos para software embarcado na industria automotiva baseada em uma arquitetura de processos de software. In Workshop Anual do MPS, Campinas.

[16] Schneidewind, L. & Hörold, S. & Mayas, C. & Krömker, H. & Falke, S. & Pucklitsch, T. (2012) “How Personas Support Requirements Engineering”, Usability and Accessibility Focused Requirements Engineering (UsARE), 2012 First International Workshop, 1—5.

[17] Silva, E. & Batista, T. & Oquendo, F. (2015) “A Mission-Oriented Approach for Designing System-of-Systems”, 2015 10th System of Systems Engineering Conference (SoSE).

[18] Silva, E. & Cavalcante, T. & Batista, F. & Oquendo, F. & Delicato, C. & Pires, P. F. (2014) “On the characterization of missions of systems-of-systems”,Proceedings of the 2014 European Conference on Software Architecture Workshops. New York, NY, USA: ACM.

[19] Vierhauser M. & Grünbacher P. (2014) “A Requirements Monitoring Infrastructure for Systems of Systems”, ASE '14, 887—890.

[20] Vierhauser M. & Rabiser R. & Grünbacher P. & Aumayr B. (2015) “A Requirements Monitoring Model for Systems of Systems”, IEEE 23rd International Requirements Engineering Conference (RE), 96—105.

[21] Yang-Turner F., & Lau L. (2011) “A Pragmatic Strategy for Creative Requirements Elicitation: From current work practice to future work practice”, Workshop on Requirements Engineering for Systems, Services and Systems-of-Systems, 28—31.

[22] Zowghi, D. & Coulin, C. (2005) “Requirements Elicitation: A Survey of Techniques, Approaches, and Tools”, Engineering and Managing Software Requirements, Springer, 19—46.

AUTHORS

Felipe Lima Duarte

Information Technology Analyst of the Information and Communication Technology Superintendence (ICTS) of the Federal Rural Semiarid University (UFERSA). He holds a degree in Computer Science from the Federal Rural Semiarid University (UFERSA) and currently holds a Master's Degree in Computer Science by UFERSA.

Angélica Félix de Castro

Graduated in Computer Science from the Federal University of Rio Grande do Norte (2000), Master in Geodynamics from the Federal University of Rio Grande do Norte (2002), PhD in Geodynamics from the Federal University of Rio Grande do Norte and Christian-Albrecht Universitat zu Kiel, Germany (2007) and Post-Doctorate in Computing from the University of Bristol, England (2015). She worked as a teacher of the Higher Magisterium at the Federal Institute of Bahia (IFBA), Campus of Vitória da Conquista, from 2006 to 2008 and is currently Associate Professor I at the Federal Rural Semi-Arid University.

Paulo Gabriel Gadelha Queiroz

He holds a degree in Computing from the Federal University of Ceará (2007), a master's degree (2009) and a doctorate (2015) from the University of São Paulo (ICMC-USP). He is currently Assistant Professor I of the undergraduate course in Computer Science at the Federal Rural Semi-Arid University (UFERSA). Has experience in the area of Computer Science, with emphasis on Software Engineering. The major research interests are: reuse, product line, Web systems, Web Services, application generators and critical embedded systems.