A Mission-Oriented Tool for System-of-Systems Modeling

Eduardo Silva¹, Thais Batista¹, Everton Cavalcante¹²

¹DIMAp, Federal University of Rio Grande do Norte, Natal, Brazil
²IRISA-UMR CNRS/Université de Bretagne-Sud, Vannes, France

eduafsilva@gmail.com, thais@ufnet.br, evertonrsc@ppgscc.ufrn.br

Abstract—Missions represent an essential information in the system-of-systems (SoS) context. A mission-based approach for specifying SoS involves the definition of the missions, the capabilities of the constituent systems required to accomplish such missions, and the interactions among these systems. Due to the fundamental role played by missions in the SoS context, such an approach could support the description of an SoS even when information related to implementation is not available. Nevertheless, despite the importance of representing mission-related information, the literature does not report specific languages, models, or tools that together enable to describe missions in the SoS context. This work presents mKAOS Studio, an open-source tool for modeling SoS missions using the mKAOS language. In this paper, we illustrate the use of mKAOS Studio with a flood monitoring SoS based on existing systems to predict and monitor floods in risky areas.

Index Terms—System-of-systems, mission, mKAOS, tool.

I. INTRODUCTION

A system-of-systems (SoS) can be understood as the result of the interaction among existing heterogeneous independent constituent systems that cooperate to form a larger and more complex system towards the accomplishment of a given mission [1, 2]. However, the result of this interaction is said to be more than the sum of the constituents as it enables the SoS to offer new functionalities that cannot be offered by any of the constituent systems as individuals, the so-called emergent behavior.

As any class of system, SoS has goals that can be described as missions. These missions can be organized in a mission level, a viewpoint of the SoS, such as required capabilities and/or desired emergent behavior, which is essential for the SoS context. Moreover, such solutions often define missions as an implementation concern, whereas a mission in the SoS context is closer to requirements [4]. Nevertheless, the existing solutions for either SoS modeling or mission modeling are not fully applicable to missions in SoS context as they require specific concepts from SoS and the notion of mission as a central element, a concern that is not provided by such solutions.

In order to address these lacks, we propose mKAOS, a mission-level modeling language that allows specifying missions of SoS and defining relationships between missions and other aspects of the SoS (such as emergent behavior and capabilities of the constituent systems), regardless of implementation details. The main goal of mKAOS is to allow a detailed modeling of missions in the SoS context, as well as to enable stakeholders to identify or define specific needs for the SoS, such as required capabilities and/or desired emergent behaviors. The language follows a conceptual model for missions of SoS introduced in our previous work [3].

This paper is focused on mKAOS Studio, an open-source tool for modeling missions in SoS by using the mKAOS language. The tool was developed using open-source modeling frameworks and is integrated with Eclipse [5], a widely used Integrated Development Environment (IDE). Aiming at illustrating the use of the mKAOS language and its associated tool, we model the missions of a Flood Monitoring SoS, which was designed upon existing systems to predict and monitor floods in risky areas.

The remainder of this paper is organized as follows. Section II presents the background of this work. Section III briefly
Section IV presents the Flood Monitoring SoS whose missions are modeled by using mKAOS Studio. Section V presents some final remarks.

II. BACKGROUND

In our previous work [3], we have outlined the elements of a mission in SoS based on an analysis of the literature about of missions in a broader context. Such a study aimed at identifying concerns that could be brought to the SoS context and other issues that should be considered when shifting from traditional, single systems to SoS. As a result, we have proposed a conceptual model for missions of SoS, as shown in Fig. 1. Some of the elements of the conceptual model were identified as essential when describing missions in the SoS context, while others are consensual in the literature, such as emergent behavior and cooperation. In Fig. 1, mission appears as a central concept correlated with five other important concepts: (i) priority, which defines the commitment of the system within the mission; (ii) trigger, which defines a necessary and sufficient condition for executing the mission; (iii) constraints, which can be classified into invariants (constraints that must be satisfied) and heuristics (constraints that are desirable, but not mandatory); (iv) parameters, which can be provided as input for a mission or be the output (result) of its accomplishment; and (v) tasks, which are functional operations that implement the mission.

It is possible to notice that some works in the literature deal with missions closely related to requirements [4, 6]. However, we argue that missions require additional information that cannot be represented through traditional requirements approaches, e.g., the interactions among constituent systems that lead to the emergent behaviors of an SoS. Nevertheless, some requirements engineering approaches can provide elements that would allow describing some of the aspects that compose a mission. One of such approaches is the KAOS (Keep All Objectives Satisfied) requirements engineering language and methodology [7]. KAOS is structured upon four main models: (i) goal model, which describes goals (i.e., primary objectives to be accomplished by the system), requirements, expectations (i.e., secondary objectives that are not under the system control), and obstacles (i.e., contexts that the system must overcome in order to achieve a given goal); (ii) responsibility model, which describes agents and assigns responsibilities, i.e., which agent is responsible for each goal; (iii) object model, which defines the objects that will be used in the system in terms of entities and events; and (iv) operation model, which is responsible for defining operations that implements the requirements.

We have noticed that some of the KAOS elements are able to represent a significant set of concepts identified in our conceptual model shown in Fig. 1. For this reason and due to the wide use of KAOS in industry, we have chosen such a language to develop mKAOS, a specialization of KAOS aimed to support the description of missions in SoS.

III. mKAOS: LANGUAGE AND TOOL

A. mKAOS Modeling Language

mKAOS allows the description of both global and individual missions by assigning individual missions to constituent systems and correlating global missions with emergent behaviors arisen from the interactions among such systems within the SoS. In addition, the language allows describing the capabilities of each constituent system. In mKAOS, we have taken advantage of the goal element of KAOS, which represents an objective of a system, to enable the description of missions since a mission in an SoS can be understood as a goal [3]. However, a mission can encompass additional information that is not originally encompassed by goals in KAOS, such as priorities and triggers.

Missions in mKAOS can be described through six different models, each one with its own syntax and semantics. These models can be overlapped aiming to provide additional, fine-grained information regarding a certain aspect of the system. In the following, we present each of these models.

Mission models. Mission models are the main type of model of mKAOS. In these models, missions are structured as tree in which leaf nodes represent individual missions and non-leaf nodes represent global missions. Refinement links establish a refinement relationship between missions, so that a given mission can be refined into other sub-missions. In addition, it is possible to define expectations.

Responsibility models. This type of model is concerned with the description of the constituent systems of an SoS, as well as with the assignment of responsibilities of each individual mission to a specific constituent system. Responsibility models can also encompass environment agents, i.e., agents that are not part of the system, but that can be agents are responsible for expectations.

Object models. This type of model is responsible for providing operational objects used by the system. mKAOS supports two main types of objects, entities and events. The former
implement a data structure that represents some element of the physical world, whereas the latter are related to specific circumstances to which the system must react.

**Capability models.** mKAOS comprises two types of capability models, namely (i) operational capability models and (ii) communicational capability models. An operational capability model defines a set of operations that each constituent system is able to execute. In these models, operational capabilities can trigger or handle events, while input and outputs for operational capabilities can be represented by entities. In turn, a communicational capability model specifies interactions between constituent systems via communicational capability elements. These elements are used when an information (or event) provided by some operational capability is required by other operational capability that is related to a different constituent system.

**Emergent behavior models.** This type of model describes emergent behaviors by using communicational capabilities. In mKAOS, an emergent behavior is a composition of several communicational capabilities. Each emergent behavior must be associated to a global mission, thus implying that each behavior is related to the achievement of a specific global mission.

### B. mKAOS Studio Tool

In order to support mission modeling with mKAOS, we have developed *mKAOS Studio*, an Eclipse [5] plug-in implemented by using facilities provided by the Eclipse Modeling Framework (EMF) [8] and the Sirius Framework [9], as depicted in Fig. 2. EMF is a framework for developing domain-specific models based on a meta-model, the so-called *ecore*. An EMF *ecore* contains a set of classes and relationships that implement an abstract language. EMF generates several artifacts representing one of the six models of the mKAOS language. When overlapping these models (i.e., their respective layers), it is possible to visualize some relationships between them and their respective elements. As an example, the Responsible For relationship, which defines the responsibility of the constituent systems on individual missions, is only visible when the mission model is overlapped with the responsibility model. In this perspective, the process of defining an SoS from its missions encompasses specifying of all models, overlapping these models, and then establishing the relationships among them.

Fig. 3 shows the main screen of mKAOS Studio. The palette at right contains the tools needed to create elements and relationships for mKAOS models. The model area (central part of the window) presents the current state of the model. Above the model area, a small toolset with tools to manipulate the model area (e.g., activating/deactivating layers, sorting elements, etc.) is available.

### IV. Example: A Flood Monitoring SoS

The flood monitoring SoS used as example in this paper is an SoS based on an existing flood monitoring system based on wireless sensor networks [10], which was adapted to present the inherent characteristics of SoS. This Flood Monitoring SoS is an acknowledged SoS whose main mission is to monitor and predict floods in a certain region. As an acknowledged SoS, goals, resources, and central control of the SoS are all recognized, but the constituent systems retain their independent management and their behavior is not strictly subordinated to the central managed purpose. Table I describes the global missions for such a system. These missions are depicted in the mKAOS model shown in Fig. 4.

The Flood Monitoring SoS results from the interaction among six constituent systems, which cooperate to accomplish the aforementioned missions. Table II briefly presents these systems. Fig. 4 also shows the representation of these systems in mKAOS and their relationship with the individual missions specified in the mission model.

Fig. 4 overlaps a mission model with a responsibility model, thus showing the Responsible For relationship. In Fig. 4,
TABLE II
CONSTITUENT SYSTEMS OF THE FLOOD MONITORING SOS

<table>
<thead>
<tr>
<th>System</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tropical Rainfall Measuring System</td>
<td>Produces rain models based on previous data and satellite images obtained by the system itself.</td>
</tr>
<tr>
<td>EO-1 Advanced Land Images</td>
<td>Refines satellite images through comparing to other images. Controls a set of drones to take land images.</td>
</tr>
<tr>
<td>Global Disaster Alert and Coordination System</td>
<td>Produces evacuation plans and triggers alerts to the authorities when informed of a natural disaster.</td>
</tr>
<tr>
<td>River Data System</td>
<td>Monitors the level of rivers and maintain fluvial models.</td>
</tr>
<tr>
<td>Civil Service System</td>
<td>Provides data related to buildings (such as their location and size) for a given region and maintain information regarding risk areas.</td>
</tr>
<tr>
<td>Geographical Model Manager</td>
<td>Produces and maintains geographical models, including soil and terrain data.</td>
</tr>
</tbody>
</table>
TABLE I
MISSIONS OF THE FLOOD MONITORING SoS

<table>
<thead>
<tr>
<th>Mission</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evacuation Plan Produced</td>
<td>To produce an evacuation plan when a flood is detected</td>
</tr>
<tr>
<td>Flood Status Monitored</td>
<td>To monitor a flood status, predicting a future flood or monitoring an in-course flood</td>
</tr>
<tr>
<td>Flood Alerted</td>
<td>To alert authorities of a detected flood</td>
</tr>
</tbody>
</table>

In the Flood Monitoring SoS, some objects are shared among constituent systems in the sense that a given system produces information that can be used by another constituent, e.g., position data (coordinates), images, and hydrological models. These objects are described in an object model, as illustrated in Fig. 5. In Fig. 5, two events are defined, Flood Detected! and Rain Detected!. Additionally, several entities are specified, each one being responsible for defining an abstraction such as a class, a datatype, or a physical object. Some entities are composite, i.e., they are composed of other entities: for instance, the River Report entity is composed of River Simulation and Authority List entities. These composition relationships are specified through composition links similar to the ones available in the Unified Modeling Language (UML). On the other hand, some entities are atomic, such as Coordinates, Image, and Hydrological Model.

Each constituent system in the Flood Monitoring SoS has its own operational capabilities, which represent how the system can accomplish its own individual missions. The operational capability model produces and/or uses the objects defined in object model as input/outputs. Fig. 6 shows the operational capability model for the Tropical Rainfall Measuring System System. In Fig. 6, the Tropical Rainfall Measuring System has four capabilities: (i) To Retrieve Satellite Data; (ii) To Report Simulation; (iii) To Simulate Rain; and (iv) To Simulate Hydrological Changes.

Communicational capabilities describe the interactions between the constituent systems. Therefore, they use the objects related to the constituent systems through their operational capabilities. Although the Flood Monitoring SoS has several communicational capabilities, it is important to focus on the interactions that somehow influence the accomplishment of the global missions. Fig. 7 shows an example of communicational capability model for the Flood Monitoring SoS. In Fig. 7, the River Levels entity is provided by To Measure River Levels, an operational capability of River Data System. This entity is used by the To Produce Rain Model operational capability of the Tropical Rainfall Measuring System. This type of interaction between the Tropical Rainfall Measuring and the River Data System is explicitly represented by the Provide River Levels communicational capability.

Finally, communicational capabilities enable the existence of emergent behaviors. There are two major emergent behaviors related to the global missions of the Flood Monitoring SoS, namely To Infer Building Data, which is related to the Evacuation Plan Produced mission, and To Monitor Real-Time Flooding, which is related to the Flood Status Monitored
global mission. Fig. 8 describes how these emergent behaviors are related to the communicational capabilities of the system. In Fig. 8, the To Infer Building Data emergent behavior is a composition of at least two Building Data Exchange and Inter-drone Data Exchange communicational capabilities. In turn, the To Monitor Real-Time Flooding emergent behavior is a composition of exactly one To Provide Hydrological Model for Simulation communicational capability and at least one of Inter-drone Data Exchange and Satellite Images to Simulation communicational capabilities. After defining the emergent behavior model, it is important to overlap it with the mission model and to define relationships between emergent behaviors and global missions.

The complete mKAOS model for the Flood Monitoring SoS and the mKAOS Studio tool are publicly available to download at http://consiste.dimap.ufrn.br/projects/mkaos.

V. FINAL REMARKS

This paper presented mKAOS Studio, an open-source mission-oriented tool for designing SoS with focus on their global missions and the individual missions of their constituent systems. The tool implements the mKAOS language, a specialization of the KAOS requirements engineering language and methodology. Mission models specified in the mKAOS language and supported by the mKAOS Studio tool can be regarded as the starting point for a mission-oriented process for developing SoS, a current lack of the literature in the SoS context. In this paper, we illustrate the use of the mKAOS language and its associated tool by modeling the missions of a Flood Monitoring SoS.

As a future work, we intend to integrate mKAOS and its tool with existing languages/tools for SoS software architecture modeling, thus enabling stakeholders to design a system in two levels, (i) the mission level, using mKAOS, and (ii) the architectural level, possibly using an architecture description language. Furthermore, it is important to enable the analysis of mKAOS models in order to verify important conditions such as “are all individual missions assigned to the constituent systems?” and “are all constituent systems necessary to accomplish the global missions of the SoS?”. Finally, we intend to: (i) validate the mKAOS language and tool in other SoS scenarios; (ii) to assess the language in terms of completeness and expressiveness for mission modeling; (iii) evaluate the mKAOS Studio with experts aiming at assessing the benefits brought by the adoption of a mission-oriented process for designing SoS.

ACKNOWLEDGMENT

This work is partially supported by CNPq and the Brazilian National Agency of Petroleum, Natural Gas and Biofuels (PRH-22/ANP/MCTI Program). The authors also thank Flavio Oquendo (IRISA/UBS, France) for the helpful comments and suggestions to improve this paper.

REFERENCES