Revisiting Goal-Oriented Models for Self-Aware Systems-of-Systems

Everton Cavalcante\textsuperscript{1,2}, Thais Batista\textsuperscript{1}, Nelly Bencomo\textsuperscript{3}, Pete Sawyer\textsuperscript{4}

\textsuperscript{1}DIMAp, Federal University of Rio Grande do Norte, Natal, Brazil
\textsuperscript{2}IRISA-UMR CNRS/Université de Bretagne-Sud, Vannes, France
\textsuperscript{3}School of Engineering and Applied Science, Aston University, Birmingham, United Kingdom
\textsuperscript{4}School of Computing and Communications, Lancaster University, Lancaster, United Kingdom

evertonrsc@pgpsc.ufrn.br, thais@ufnet.br, nelly@acm.org, p.sawyer@lancaster.ac.uk

Abstract—Systems-of-systems (SoS) are systems resulted from the interaction among other independent constituent systems that collaborate to offer new functionalities towards accomplishing global missions. Each of these constituent systems accomplishes its individual missions and is able to contribute to the achievement of the global missions of the SoS, both being viewed as a set of associated goals. In the perspective of self-aware systems, SoS need to exhibit goal-awareness, i.e., they need to be aware of their own goals and of how their constituent systems contribute to their accomplishment. In this paper, we revisit goal-oriented concepts aiming at identifying and modeling goals at two levels, namely the SoS level and the constituent systems level. Moreover, we take advantage of such goal-oriented models to express the relationship among goals at these levels as well as to define how each constituent system can contribute to the accomplishment of global goals of an SoS. In addition, we shed light on important issues related to goal modeling in self-aware SoS to be addressed in future research.

Keywords—self-awareness; systems-of-systems; requirements engineering; missions; goals; goal-awareness.

I. INTRODUCTION

A system-of-systems (SoS) can be defined as the result of the interaction among other distributed, independent constituent systems that cooperate to form a larger and more complex system towards accomplishing global missions [1]. Each of these heterogeneous constituent systems has its own individual missions and can contribute to accomplish the global missions of the SoS. Such a collaboration is said to be more than the sum of the constituent systems as it enables the resulting SoS to have an emergent behavior related to new functionalities that cannot be individually provided by any of these systems. Besides emergent behavior, SoS present an evolutionary development: they can evolve due to changes related to the environment, to their constituent systems, or to their functions and purposes. Furthermore, constituent systems within an SoS have operational and managerial independence, i.e., they can provide their own functionalities and be managed independently from the SoS.

In another perspective, self-aware systems have been usually defined as computing systems that exhibit a set of three fundamental properties: (i) self-reflection, which refers to the ability of a system to be aware of its software architecture, execution environment, and hardware infrastructure on which it is running as well as of its operational goals and quality requirements; (ii) self-prediction, which enables the system to predict the effect of dynamic changes and possible adaptation actions, such as adding and/or removing resources; and (iii) self-adaptation, which refers to the ability of a system to proactively adapt as the environment evolves in order to ensure that its operational goals are continuously met. Considering the inherent characteristics of SoS, it is possible to notice that this new class of systems can highly benefit from these self-awareness properties, leading to the emergence of self-aware SoS (also known as collectively-aware systems).

In SoS, both global and individual missions can be viewed as a set of associated goals. Despite goals represent the main concern addressed by traditional requirements engineering techniques, these approaches cannot be applied for SoS as they successively decompose a system in a bottom-up direction, i.e., from requirements to component systems. For this reason, they are not able to capture important concerns related to the constituent systems and how they influence the overall SoS in a bottom-up direction, e.g., how constituent systems contribute to the realization of the global goals established for the SoS. Nevertheless, these techniques cannot be completely disregarded as they are able to provide relevant insights towards an approach suited to SoS [2].

In this paper, we particularly focus on the goal-awareness facet of the self-reflection property by arguing that a self-aware SoS needs to be aware of its own goals and of how constituent systems contribute to their accomplishment. First, we briefly discuss the motivation for an SoS to present the aforementioned self-awareness properties in the light of the characteristics inherent to this class of systems (Section II). Next, we revisit goal-oriented concepts aiming at considering both SoS and constituents perspectives to: (i) model goals of SoS and constituent systems; (ii) express relationships between goals at the SoS level and the constituent systems level; and (iii) define how each constituent system contribute to the accomplishment of global goals of an SoS (Section III). Finally, we present some concluding remarks and directions for future work (Section IV).

II. SELF-AWARENESS AND SO S

SoS and self-reflection. The concrete constituent systems that compose an SoS at runtime are often partially known
or even unknown at design time. As a result, these concrete constituents must be discovered, selected, and composed at runtime in order to identify proper arrangements of these systems that contribute to the accomplishment of the global goals of the SoS based on their capabilities. Furthermore, due to their operational and managerial independences, constituent systems within an SoS can present a life cycle on their own, i.e., they have their own missions, operate independently of the SoS, and can be managed and evolved independently from the other constituents [1]. In another perspective, it is important to consider the degree of coercion enforced by the SoS over its constituents and that may affect their operational and managerial independences. For example, an SoS may have constituents that are free to operate independently, while another SoS could have constituents that are obliged to operate within its scope despite the operational independence of the constituent systems. Finally, an SoS need to present goal-awareness, i.e., it needs to be aware of its own goals as well as awareness of its independent systems and how they are composed together to meet such goals. Therefore, SoS can clearly benefit from self-reflection as they need to continuously reason about itself to cope with these multiple facets. In addition, self-reflection in SoS seems to be quite useful to guide the interactions among constituent systems towards the achievement of global goals.

**SoS and self-prediction.** One of the main features that makes SoS distinct from other classes of systems (and at the same time the most difficult one to handle) is the so-called emergent behavior. In fact, an SoS depends on emergent behaviors to achieve its purposes as they enable it to provide new functionalities resulting from the collaboration among its constituent systems. Some emergent behaviors can be predicted, foreseen, so that they can be determined by specifying interactions between constituent systems. On the other hand, emergent behaviors can be also unpredicted, unforeseen, thus dynamically appearing in the context of the SoS. Moreover, both foreseen and unforeseen emergent behaviors can be desirable or even undesirable, i.e., the result of the interactions among constituent systems within an SoS can be respectively positive or negative over its operation [3]. Due to these different natures of emergent behavior that may occur within SoS, these systems need to present self-prediction capabilities in order to foster desirable behaviors and mitigate the undesirable ones, as well as to assess their impact in its operations. Self-prediction is fundamentally important for critical-mission SoS, in which a potential financial, personnel, and environmental impact exists.

**SoS and self-adaptation.** As previously mentioned, SoS are inherently dynamic in terms of the ability of changing their structure, functions, and purposes at runtime. Therefore, an SoS must remain operational upon environmental changes and/or changes in its constituent systems. In this perspective, self-adaptation capabilities of SoS can enable these systems to reason about themselves and adapt autonomously to function continuously and/or achieve particular quality objectives in the face of uncertainties and changes. However, the operational and managerial independences of constituent systems within an SoS pose challenges related to self-adaptation: such systems are autonomous entities that may evolve by their own without the control of the SoS and hence may impact the overarching structure. Furthermore, the inherent geographical distribution of constituent systems within an SoS hampers the adoption of centralized solutions for controlling self-adaptation, mainly when taking into account quality requirements such as resilience, robustness, and scalability [4]. Despite the relevance of self-adaptation as a key concern in SoS, there has been a limited work paying attention to it [5, 6]. Until now, self-adaptation is the only self-awareness property addressed in the literature.

### III. Goal-Oriented Modeling of SoS

At a high-level, requirements engineering activities for SoS should consider both perspectives regarding constituent systems and the SoS itself. However, a top-down approach (from the SoS to constituent systems) used in isolation might not be able to adequately consider aspects related to the constituent systems. In turn, a bottom-up approach (from constituent systems to the SoS) may not be able to capture important concerns related to the SoS as a whole. Therefore, we argue that a successful requirements engineering approach for SoS would require a combination of both approaches. On the one hand, top-down aspects approach the understanding of the SoS, the environment(s) associated with the SoS and its constituent systems, the identification of goals, and the identification of the SoS interactions. On the other hand, bottom-up aspects include a proper understanding of the individual goals of the constituent systems and the capabilities that they provide, as well as internal and external constraints to be considered.

One of the most important steps within requirements engineering for SoS is the identification and specification of the goals to be accomplished by the SoS and the individual goals of the participating constituent systems. Inspired in the Lewis et al.’s work [2], we propose an approach structured upon two goal levels for modeling goals in the SoS context. The first one is the SoS Goal Level, which encompasses the representation of global goals of the SoS itself. In turn, the second level is the Systems Goal Level, which encompasses the representation of goals of the individual constituent systems. With this approach, it is possible to express the global goals of the SoS as well as to handle the collaboration of independent systems and their respective goals, mainly in terms of how goals specified at the Systems Goal Level contribute to accomplish the global goals specified in the SoS Goal Level. Figure 1 depicts these two goal levels for modeling goals of a SoS and of its constituent systems. Goals $G_1$ and $G_2$ represent the global goals of an SoS, which
can be decomposed (refined) into sub-goals ($G_1'$ and $G_1''$) at the SoS Goal Level. Moreover, goals $Cs_1$, $Cs_2$ and $Cs_3$ at the Systems Goal Level are associated with the constituent systems that collaborate for achieving the goals of the SoS. In Figure 1, individual goals $Cs_1$, $Cs_2$ and $Cs_3$ respectively contribute to global goals $G_1'$, $G_1''$ and $G_2$.

In order to model goals of a SoS and of its constituent systems, we take advantage of goal-oriented requirements engineering (GORE), a successful approach that has been used along the years and that considers goals as drivers for requirements elaboration as they describe what a system is designed to achieve. The literature presents several well-known frameworks and notations for modeling goals of a system, such as KAOS [7], $i^*$ [8], and GRL [9]. Apart from the existing differences among these approaches, they share a set of main common concepts, namely: (i) actors (or agents), which are active entities representing a stakeholder and/or a system whose goals are to be achieved; (ii) goals, which are functional objectives of actors that can be fully achieved; (iii) softgoals, which refer to objectives that may not be fully satisfied; and (iv) links, which allow establishing different relationships among actors, goals, etc.

GORE is a promising, effective approach in the SoS context as goals can be considered as the main starting point to requirements engineering for SoS [10]. In this perspective, goal-oriented models for SoS can be inspired in the elements defined in traditional GORE approaches: actors can be used to represent the SoS itself and its constituent systems while goals can be used to represent both global and individual goals. Nevertheless, there are important specificities related to the SoS context that are not properly addressed by the set of concepts encompassed by the aforementioned approaches, as we briefly discuss in the following.

A. Links within/between goal levels

As an example, $i^*$ and GRL notations provide contribution and decomposition links to express relationships among intentional elements (such as goals and softgoals): the former can be used to define the quantitative/qualitative impact of how an intentional element contributes to the satisfaction of another one, whereas the latter allows decomposing an intentional element into sub-elements. In goal-oriented models for SoS, these links can be used mainly for expressing relationships between goals within both SoS Goal Level and Systems Goal Level. For example, a given goal in the can be decomposed (refined) into sub-goals that realize such a parent goal once they are satisfied. Similarly, it is also possible to model how a goal quantitatively and/or qualitative contribute to the realization of another one.

Contribution links can be also used to model relationships between goals at different goal levels, i.e., to express how the individual goals specified at the Systems Goal Level contribute to the realization of the global goals specified at the SoS Goal Level. In this perspective, the type (whether positive or negative) and the quantitative amount of such a contribution can be specified, thus enabling to represent full and partial contributions of a goal of a constituent system towards the realization of a goal of the SoS. These links are relevant for modeling the awareness of the goals of the constituent systems with respect to the SoS.

B. Expressing emergent behaviors

An SoS depends on emergent behaviors to achieve its purposes and to provide new functionalities resulting from the collaborative work of its constituent systems. This brings up three important issues. First, representing emergent behaviors is not a trivial task as they are essentially associated to the interactions among the constituent systems, i.e., they only appear when such systems interact within the scope of the SoS. Second, some emergent behaviors can be foreseen, so that they can be determined by specifying interactions between constituent systems or representing interaction patterns, i.e., the ways in which they can interact. Finally, emergent behaviors can be also unforeseen, thus dynamically appearing in the context of the SoS.

In order to address these issues, a new type of link called interaction link can be created to handle the explicit representation of emergent behaviors and to express how the interaction between two actors (constituent systems) yields new functionalities (goals). Therefore, an interaction link can be viewed as a relation connecting at least two actors (systems) and at least one goal to represent that such a functionality results from the interaction between these constituent systems. In the case of foreseen emergent behaviors, interaction links can be predicted by using simulation or probabilistic techniques, thus enabling their explicit representation in the goal model. On the other hand, as unforeseen emergent behaviors are dynamic, the goal model also must address these dynamic interactions as well as the resulting goals. Finally, both foreseen and unforeseen emergent behaviors can be desirable or even undesirable, i.e., the result of the interactions among constituent systems within an SoS can be respectively positive or negative over its operation. As a result, interaction links must also support representing the impact type (positive/desirable, negative/undesirable) and amount of such interactions.

Figure 1. Goal levels for goals of SoS.
IV. Final Remarks

In this paper, we highlighted the relevance of self-awareness properties for SoS, a new class of systems resulted from the interaction among independent constituent systems towards accomplishing global goals. Specifically, we have focused on the goal-awareness facet of the self-reflection property as means of enabling an SoS to be aware of its global goals and of how its constituent systems (with individual goals on their own) collaborate towards achieving them. In this context, we revisited goal-oriented concepts for modeling goals within SoS and for expressing relationships among goals at both SoS and constituent systems levels. Additionally to the elements provided by well-known goal-oriented approaches existing in the literature for traditional systems, we introduced the notion of interaction links aiming at properly encompassing emergent behaviors in SoS, which refer to functionalities resulted from the interactions among constituent systems and related to the goals to be achieved by the SoS. Despite the agreement on the fundamental role played by goals in this scenario, there are just few proposals for modeling goals of an SoS and its constituents [11].

An important issue in the SoS context is addressing which concrete constituent systems should be integrated into the overarching SoS. This is an important activity as an SoS may emerge with partial or even no knowledge about the concrete systems that will be integrated to compose it. To deal with this issue, the self-reflection property can be useful as means of analyzing goal models towards the selection of the constituent systems able to accomplish the global goals of an SoS. Such analysis and selection processes can indicate how alternative architectures are feasible for a self-aware SoS by assessing impact of selecting a given candidate system over another considering goal satisfaction.

In the SoS context, conflicts among goals may arise for a variety of reasons, such as: (i) the existence of a broader range of stakeholders including both stakeholders of the individual constituent systems and the stakeholders of the overall SoS, each one with different interests in the SoS; (ii) conflicts in the relationship between constituent systems and the SoS due to their managerial independence; (iii) conflicts arisen from interactions among constituent systems due to their operational independence; and (iv) the fact that a given constituent system might simultaneously belong to more than one SoS. Therefore, it is necessary to provide strategies and solutions to tackle these different scenarios that may result in conflicting goals and to assess the impact of such conflicts in the operation of the SoS and its constituents.

Finally, inherent features of SoS such as evolutionary development require addressing important concerns regarding requirements engineering and goal modeling in this context. Uncertainty [12] is a concern that must be handled in this context mainly due to the dynamic, unforeseen changes in the operation environment in which SoS are inserted and to emergent behaviors (some of them unforeseen) arisen from interactions among constituent systems. Another implication of the dynamic evolution of SoS is addressing goals as runtime entities [13], thus allowing monitoring, reasoning, and tracing changes on such goals during the SoS execution.

REFERENCES