Situation/Reaction Model For Systems of Systems construction

Rymel Benabidallah[1], Salah Sadou[2], and Mohamed Ahmed Nacer[1]

[1] University of Science and Technoloy Houari Boumedienne, Algiers, Algeria
rbenabidallah@usthb.dz, anacer@cerist.dz
[2] University of South Brittany and IRISA, Vannes, France
Salah.Sadou@irisa.fr

Abstract

The construction of systems of systems (SoS) has become the new challenge in the field of complex systems. One of the SoS's properties is the dynamic adaptation to changes in their environment. This implies a permanent on the fly rearchitecturing of the SoS. Thus, one may doubt about the usefulness of defining the first architecture of the SoS and consequently the role of the architect.

Through this paper, we argue that the design can be focused on the most stable part of SoSs: their mission. Thus, we propose an approach of constructing SoSs through the definition of the mission. The definition of the mission is based on the principle of situation/reaction with an implied architecture. We propose a definition model where the dynamic aspect of the SoS's environment is implicit. The situations are defined by an expert system and reactions by orchestrations. The aim of this paper is to present a meta-model allowing to describe SoS based on their mission.

1 Introduction

Progressively almost all aspects of our lives and livelihoods depend on some kind of software-intensive system. With this profusion of systems and means of communication, it would seem more appropriate to combine (or make collaborate) existing systems to meet the new need. The result of this collaboration is called System of Systems (SoS) [2]. SoS is an emergent domain in the research community. The aims of most of work on SoS target developing novel theories and technologies for architecting and engineering them in order to cope aspects such as dynamic evolution and security [6].

One of the significant aspects and also inherent in the actual nature of the SoS is the dynamicity [2, 7]. Indeed, a SoS uses existing systems within a dynamic environment: the used systems can disappear and new ones may appear. Thus, the architecture of a SoS must be perpetually and dynamically adapted to these changes in order to take into account and take advantage from the new state of its environment. Hence our question: what is the point of describing the architecture of a SoS whether it should be perpetually changed?

To solve the problem related to dynamic changes of SoS's architecture, it is possible to continue to use a purely architectural approach. Indeed, it is possible to define a generic architecture associated with mechanisms and policies for the adaptation. But we believe that with the SoS the architecture is no longer the backbone of any successful software-intensive system [1]. Indeed, this assertion is based on the fact that the software architecture is the most stable element in a system. But this is not the case for SoS, as we have just shown.

In this paper we propose to consider the mission as stable element during the construction of a SoS. Indeed, a SoS is built to carry out a mission. So the approach we propose puts the focus on the design of the mission, whereas until now the design concerned directly the constituents

of the systems. The concrete solution will be generated from the definition of the mission based on a generic architecture which is not influenceable by changes in the environment. The definition of the mission is based on the Situation/Reactions paradigm. We consider that a SoS is associated with an environment consisting of all possible actors (element types) that may affect the mission. The elements of the environment can be of different nature: software intensive systems, natural elements (such as weather or obstacles) and even humans. The paradigm of Situation/Reactions allows to define the mission in the form of collaborations that the actors must achieve in case of a particular situation detected in the environment.

To achieve this SoS building approach, we propose a meta-model for modelling the mission and its environment, the use of first order predicate to define situations, a basis of facts for the representation of the current state of the environment and finally, the representation of reactions using orchestrations. The basis of facts will be fueled by elements of the environment and the SoS itself. With this approach of SoS building, architecture is implicit and the dynamic adaptation to changes in the environment is achieved at the predicates level.

The rest of this paper is organized as follows: we describe the proposed paradigm and approach for SoS construction in the next section. In Section 3, we describe the meta-model for modelling the SoS and its environment. We describe a case study in Section 4 and conclude the paper in Section 5.

2 General Approach

Engineers from system engineering domain are faced with problems concerning SoS since a while. The concept of SoS's mission is central during the specification of their SoS. Unfortunately, the literature in software engineering domain lacks of researches on the specification of this notion.

Unlike in the field of engineering system in SoS software engineering field consist of volatile subsystems. Thus, the most important characteristic to take care when designing an SoS is its ability to adapt to its environment. As sketched in Figure 1, to ensure the adaptability of the SoS to its environment, the process we propose begins by describing the context/environment in which the SoS operates. Thus, each element related to the SoS will be completely specified. Element's state changes are defined as facts that may influence the way to accomplish the mission. Situations are expressed as logical rules using facts and are conditions for the state changes, role Assignment or trigger of activity orchestrations.

The fact description basis cover the environment's elements and the existing systems with their properties. The current values of facts corresponds to at least one of the conditions to engage facts' value change or to a trigger of an activity. In case of triggering an activity, the rule basis is consulted to select the needed rules. Indeed the activities need an executor that cannot be invoked before the role selection. By consulting the role selection Rules, one or several roles can be solicited. Roles represents the connections between the desired goal and the possible actors (constituent systems). Our approach aims at separating the constituent systems from their roles in the contribution to the mission in order to achieve the SoS dynamicity. Constituent Systems involved in a SoS are only considered as an executor instances, which are assigned to roles based on their availability and skills making them candidates for a vacant role. So the static part in the mission is "Role part" that will be assigned in a dynamic manner to the actual executor.

All these notions with their relationships define the mission Meta-model described in the next section.

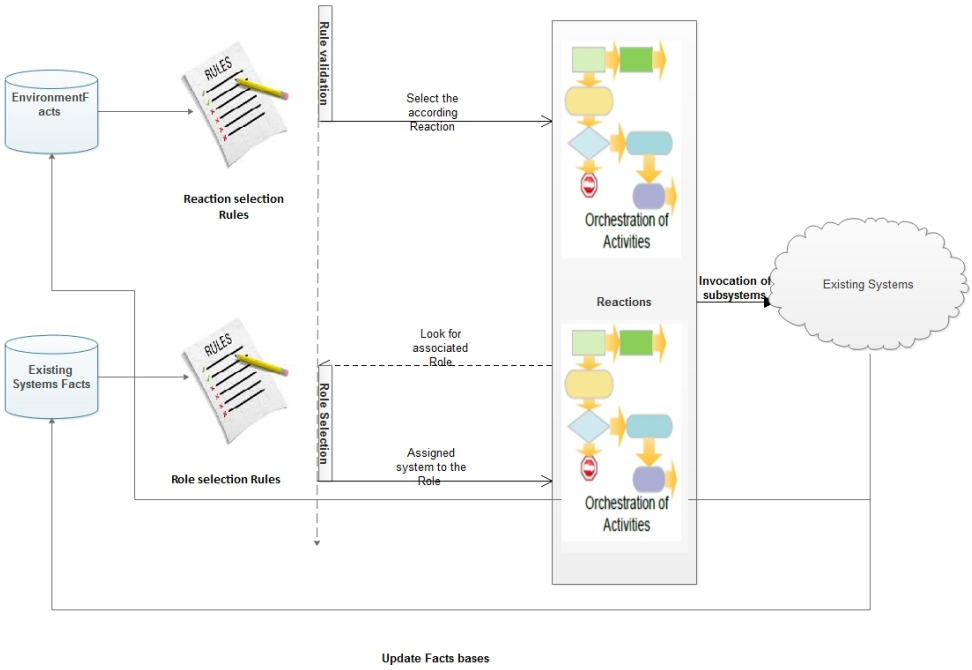

Figure 1: Process of the Approach

3 Mission Metamodel

Through the mission meta-model (see Figure 2) we describe the whole environment where the SoS accomplishes its mission. The building of SoS is based on the concept of mission (meta-class Mission), while respecting the characteristics of SoS that make it different from the other monolithic systems.

The definition of a mission is obtained by the combination of two main concepts: Role and Reactivity. Let us explain our meta-model through these two concepts

3.1 Concept of Role

The collaboration of the constituent systems produces an emergent behaviour. In our approach, this emergent behaviour is reached through the assignment of a set of roles (meta-class Role) to the appropriate actors (meta-class Constituent_System). The operation of the assignment depends on the prerequisite competencies (meta-class Competency) that the constituent systems have. The role aspect is a part of the meta-model is represented by the assignment of the constituent systems to the Roles. The role assignment is a repetitive process, it is solicited at the beginning of the mission and each time when a constituent system leave the SoS. The process consists at finding the next appropriate candidate for the vacant role. Choosing the right candidate is based on comparing the skills (attribute of the meta-classes Role and Constituent_System) possessed by the constituent system and those required for the role in question. But sometimes more than one constituent system can stand for the vacant role. In this case we need to compare the competencies of each of the candidates. This comparison is accomplished on the basis of the *weight property* the needed competencies.

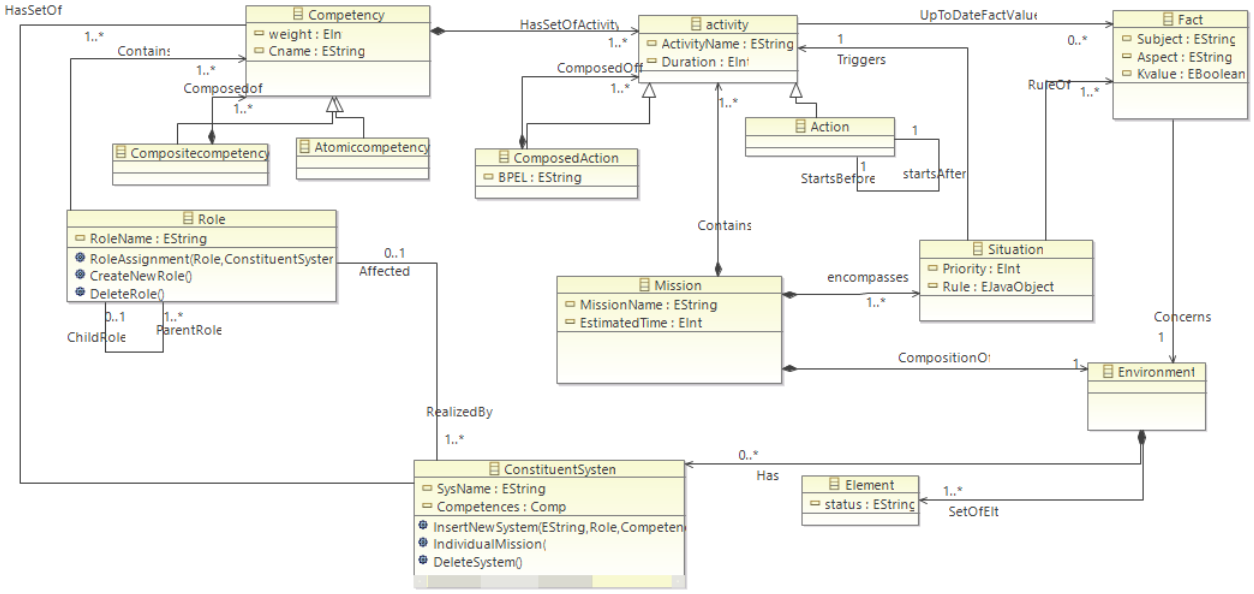

Figure 2: SoS's Mission Metamodel

3.2 Concept of Reactivity

The reactivity of the SoS is provided by the ability of the constituent systems to act upon identification of certain situations generated by the actors of the environment (meta-class Environment).

We have to notice that it exists a mutual influence between the mission and its environment. The elements (meta-class Element) of the environment can affect positively or negatively the mission of the SoS. During their life cycle, they provide information about their changes forming with a basis of facts (meta-class Fact). Some combinations of facts represent a state of the environment which corresponds to an important situation for the mission. The identification of such situation is achieved thanks to some conditions on the state of the environment (meta-class Situation). Such situations need the constituent systems to react through an orchestration of activities (meta-class Activity).

Actually, several situations may occur in the same time raising the following question: which one to deal with first if they cannot be handled in parallel. Some situations are qualified as critical, such as those that can lead to death in the medical field. That is why it is essential to detect these critical situations and distinguishing them from the other. To this aim we introduced the notion of priority (attribute of meta-class Situation) that must be fixed for each situation. As situations are discovered by conditions, so the priority is related to the order of evaluation of the conditions.

4 Telediabete Case Study

To check the consistency of our meta-model, we have modelized a SoS representing a telediabete system.

We have elaborated the specification of such a system on the basis of information taken from several research papers in computer science and medicine fields [3, 4, 5]. The system is intended to help patients with diabetes manage their chronic disease daily. The model is represented by Figure 3 in an illustrative way in order to facilitate its understanding.

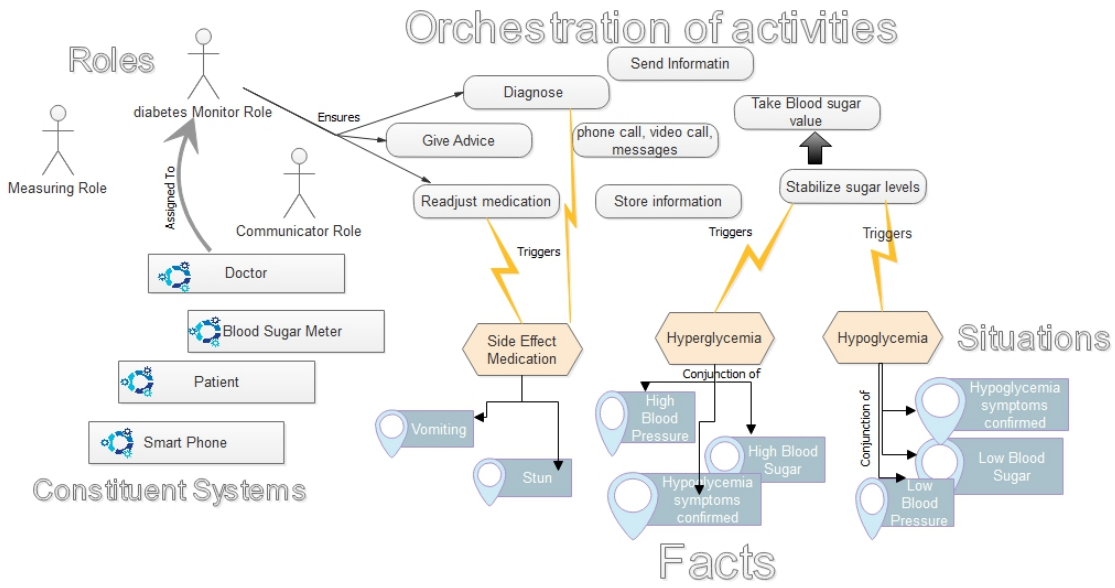

Figure 3: Telediabete System of Systems

Regular self-monitoring of blood glucose combined with appropriate action in response to blood sugar readings was thought to be the key to control the evolution of the disease. According to [3] Two complementary processes define Telediabete:

- **Telemonitoring** allows doctors remotely monitoring between two medical visits. This includes the recording, transmission and display of main variables considered in diabetes care. Variables are: glucose levels, pressure, the insulin dosage, diet, and physical activity.
- **Telecare** allows doctors remotely taking care of the patient, and the patient to ask for advice on daily situations. This service may include two supplements activities:

  Teleconsultation: exchange of asynchronous messages between the doctor and his patients.

  Supervised care: self-care/self management requires the ability to share accurate and consistent data so that the care team can assist patients in making informed decisions, provide earlier intervention to enhance quality of life, and prevent complications.

Constituent systems are not the lonely actors in the Telediabete SoS. Other elements of the environment which can not be classified as systems, such as Doctor or Patient, can affect the progress of the mission. During their life cycle, both elements and constituents provide information about their changes Fuelling thus the fact basis.

4.1 Implementation

To construct an SoS using our approach we need to express on a one hand the situations and on the other hand the reactions. Here we describe how to implement these two aspects.

4.1.1 Expressing Situations

Various knowledge representations exist in the computer science field, but the most popular form is the "*If-Then*" rule. Knowledge represented as If-Then rule is easily understandable. An other cause for the popularity of this representation form is the facility of implementation; new knowledge can be easily added, and existing knowledge can be changed simply by creating or modifying individual rules [8]. An expert system is used here to describe elements taking part in the mission's universe and spot their changes. A rule states a relationship between assertions or facts and, depending on the situation, can be used to generate new information or prove the truth of an assertion. Differently from usual uses of an expert system, we will not limit the rules processing to produce new facts but to associate a reaction to the situation described by the fired rule.

A rule is constructed by the administrator of the SoS, as the conjunction of several assertions (variables with a precise values). Bellow we give an example of such rules:

Rule 1: If HyperglycaemiaSymptoms(yes) & BloodSugar >= 10.0 mmol/L 2 hours after meals Then Hyperglycaemia (yes).
Rule 2: **If** Hyperglycaemia(yes) **Then** Invoke HyperglycaemiaReaction().
Rule 3: **If** HypoglycaemiaSymptoms(yes) & BloodSugar <= 4.0mmol/L **Then** Hypoglycaemia (yes).
Rule 4: **If** Hypoglycaemia(yes) **Then** Invoke HypoglycaemiaReaction().

From the rules above we can distinguish two types of rules. The first one (Rule 1 and Rule 3) has a new fact to inject to the Fact basis as a consequence if all the assertions are true. The second one; (Rule 2 and Rule 4); if the first part of the rule is true the consequence is an orchestration of activities.

It is obvious that all the rules can't have the same importance in the mission, some of them will reflect critical situations that should be treated first *priority*. In the expert system, priority of situations is applied as the order in which the rules are classified and then consulted in the rule basis. If two rules or more have the same priority, the order is not important especially if they are related to activities executed by separate Constituent systems.

The administrator of the SoS can ensures the role assignment process by inserting a new type of rules. Rules bellow represents the assignment of roles to the adequate subsystem.

Rule 5: If Competencies(Role) $\subset$ Competencies (Subsystem) Then Candidate(Subsystem, Role)=yes
Rule 6 : If Subsystem.Availability = yes Then invoke Assign(Subsystem, Role)

Rule 5 consists in comparing the set of competencies offered by a subsystem to the set of competencies prerequisite to the role. For example to make a phone call, as a result to Rule 5, we can have two participant candidates to the role of communicator: Phone or Cell Phone. Rule 6 assigns the appropriate subsystem to the role depending on the availability of the subsystems.

4.1.2 Expressing Reactions

We use orchestrations to implement reactions. An activity orchestration involves the participation of one or several roles depending on the competencies needed for each activity. Let us take our example to explain the implementation of reactions as orchestrations (also sketched in figure labelHypo).

Two principle critical situations can occur in the diabetic lifetime, Hyperglycaemia and Hypoglycaemia. A precise orchestration of activities is defined for each of these situations and

need the contribution of all or part of the actors. The reaction of the telediabete SoS to an Hyporglycaemia is achieved thanks to the contribution several roles. First of all, *Communicator Role* is used to contact emergency service in case that the patient is unconscious. Next step aims at getting out the patient from the critical state by taking fast acting sugar and space out glucose level tests. This needs the participation of the bellow roles, that can be assigned to an exclusive subsystems:

- *Measuring Role*: this role consists in measuring blood sugar.
- *Patient Role*: it verifies if the level of blood sugar is low, and use Alarm Role to space out the blood sugar measuring and taking fast acting sugar.
- *Alarm Role*: is used to mention mealtime and time of treatment.

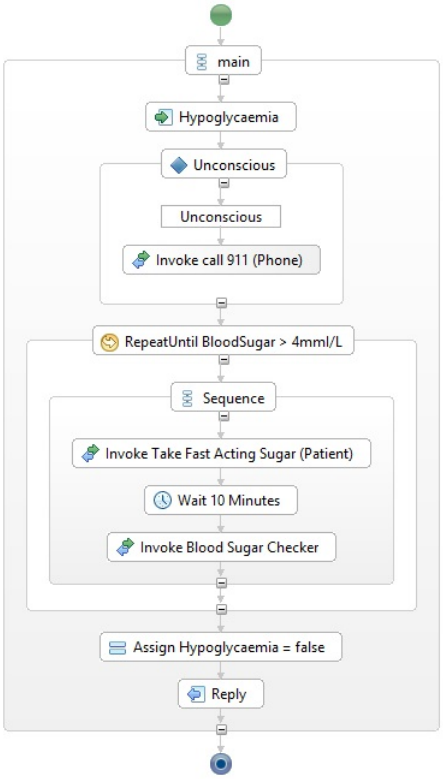

Figure 4: Hypoglycaemia situation's Reaction

In this example the Hypoglycaemia fact has new value once the critical situation has been passed. This will induce a new chaining of the rule basis and the hole process of firing rules and triggering orchestrations will start again. After executing a reaction, it is obvious that new facts are injected to the knowledge basis reflecting the new state of the mission's universe.

5 Conclusion

In this paper we presented the construction of system of systems that continuously deal with dynamic changes. A metamodel is proposed for the specification of an SoS on the principle of Situation/reaction. Through an example, simplified here for reasons of space limitations, we showed some the expressiveness level of this meta-model.

Through the use of an expert system to express all the knowledge surrounding the SoS's mission and activity orchestration making the SoS react in order to reach its targeted mission, we allow dynamic adaptation of the SoS to changes in its environment.

As future work, we target the implementation of other SoS scenarios in order to validate and/or complete our meta-model and its related processes.

References

[1] Len Bass, Paul Clements, Rick Kazman. Software Architecture in Practice. Addison-Wesley, 25 sept. 2012 - 624 pages

[2] Maier, M. W. (1996, July). Architecting principles for systems of systems. In INCOSE International Symposium (Vol. 6, No. 1, pp. 565-573).

[3] Paul Chaulk, MSc assisted by Shauna Fuller, MA, Atlantic Evaluation Group Inc. An Evaluation of the Telediabetes Report Monitoring Initiative.

[4] Van Bemmel JH, Musen MA (1997) Handbook of medical informatics. MIEUR, 1999.

[5] American Diabetes Association. "Standards of medical care in diabetes2013." Diabetes care 36.Suppl 1 (2013): S11.

[6] Jamshidi, M. (Ed.). (2011). System of systems engineering: innovations for the twenty-first century (Vol. 58). John Wiley & Sons.

[7] Firesmith, D. (2010). Profiling systems using the defining characteristics of systems of systems (SoS).

[8] Joseph P. Bigus, Jennifer Bigus. Constructing Intelligent Agents with JavaA Programmerss Guide to Smarter Applications. 14th European Meeting on Cybernetics and Systems Research-EMCSR, Toronto, 1998.