Towards the Dynamic Evolution of Context-based Systems-of-Systems

Elisa Yumi Nakagawa[1], **Rafael Capilla**[2], **Francisco J. Díaz**[3], **and Flávio Oquendo**[4]

[1]University of São Paulo – USP, São Carlos, Brazil

[2]University Rey Juan Carlos – URJC, Madrid, Spain

[3]Ingeniería y Economía del Transporte – INECO, Madrid, Spain

[4]IRISA - University of South Brittany – USB, Vannes, France

elisa@icmc.usp.br, rafael.capilla@urjc.es,
fjdiaz@ineco.com, flavio.oquendo@irisa.fr

Abstract. *Systems engineering has invested considerable efforts in the development of complex-large systems, often known as Systems-of-Systems (SoS). However, the systematization of current engineering practices based on the inherent complexity and size of these systems is still challenging for software engineers. In this light we focus in this paper on the evolution aspects of SoS using dynamic variability techniques. We suggest paths where runtime variability mechanisms can help to manage the evolution of such complex-large systems.*

1. Introduction

The discipline of Systems Engineering is considered an interdisciplinary field focused on the design and management of large-scale systems [Sage 2000]. Systems engineering deals with all those organizational and technical disciplines where a complex system is considered and integrated as a whole. System engineering overlaps with other technical and human-centered disciplines (e.g., control engineering, project management, industrial engineering or mechatronics) aimed to produce and manage complex software systems. Nowadays, the genesis and development of large Ultra-Large-Scale (ULS) systems [Northrop 2006] perceived as a System-of-Systems (SoS) bring many challenges from its conception to its maintenance and evolution.

SoS encompasses the integration of various operationally independent systems, even developed with different technologies and for diverse platforms. An adequate integration has been more and more necessary to promote cooperation among these independent systems in order to provide more complex functions, which could not be provided by any system working separately [Maier 1998]. One important challenge for SoS development refers to its evolutionary development. Complex systems that exploit context-awareness and modify their behavior at runtime demand more complex maintenance procedures. As functions and purposes of SoS can change at runtime and new constituents can be reassembled to perform a different mission, the challenge for adaptive behavior in SoS demands specific solutions.

In this scenario, this paper discusses about SoS evolution. In particular, we focus on addressing how dynamic variability techniques can support the evolution of such complex systems. The remainder of this paper is as follows. In Section 2 we characterize SoS from an architectural point of view. Section 3 discusses the evolution and needs of SoS that demand dynamic adaptation and Section 4 motivates the role of dynamic variability as key for the evolution of SoS. In Section 5 we outline the application domain of Airport Management Systems and in Section 6 we suggest how dynamic variability can help to develop and evolve better such large-scale systems. Finally, in Section 7 we draw our conclusions and future work.

2. SoS Characterization

An SoS is constituted of various operational and managerially independent, heterogeneous constituents interoperating and complying with a larger mission [DoD 2008]. Several examples of SoS can be identified; for instance, a medical system integrating systems to diagnosis, treatment, and management of patients, airport system, and avionics system, including several of them characterized as critical embedded systems.

Two main characteristics are directly related to the constituent systems of an SoS: (i) Operational independence: each constituent can deliver its own functionalities when not working with other constituents; and (ii) Managerial independence: constituents keep their own managerial sphere, being sometimes developed by diverse companies or other institutions. In addition, considering the whole SoS, four characteristics are inherent: (i) *Emergent behavior*: SoS can deliver new functionalities resulting from constituents working together; (ii) *Evolutionary development*: functions and purposes of SoS can dynamically change at runtime; (iii) *Geographic distribution*: constituents of an SoS can be geographically distributed; and (iv) *Runtime architecture*: a new organization of the constituents is sometimes necessary in order to comply with the SoS evolution. As result of these characteristics, SoS becomes naturally unpredictable regarding to what can happen during their execution. Moreover, unpredictability also characterizes the environment where a SoS is being executed. Moreover, software essentially influences the design, construction, deployment, execution, and even evolution of SoS. Therefore, the backbone of the characteristics of both the constituents and the whole SoS is the software-intensivity.

Considering the rise of software-intensive SoS and intending to systematize their development, several initiatives from Software Engineering area can be identified. Both academy and industry have invested efforts in that direction; however, in general, these initiatives need still to be more experimented and matured. From the SoS software architecture perspective, it is also required more research efforts. Software architectures promote quality attributes, mainly interoperability, adaptability, security, reliability, and performance, considered most relevant ones for SoS [Nakagawa 2013]. There are also challenges that need be overcome, including the investigation and establishment of approaches to create, represent, evaluate, and especially evolve these architectures. Specifically, these architectures must be also prepared to support dynamic evolution, as evolution has sometimes occurred without adequate, systematic control, resulting in degradation of their quality.

3. SoS Evolution Challenges

Managing the complexity of SoS is not easy, demanding complex organizational procedures. Several modern systems use context information to more efficiently perform the possible variations and adaptation of their complex operations, which require changes in their behavior. As the evolution of SoS is naturally a challenging, complex task, SoS should optimally and adaptively manage the information and resources according to varying context conditions (e.g., smart cities, intelligent transportation, wireless sensors, or warfighter systems).

Viewing SoS as a software product line where many systems can interoperate to achieve the desired functionality facilitates the development, evolution, and management of SoS. As the variety of SoS can be huge, we will focus on this work on those systems that require adaptive behavior and where the integration of complex technologies and platforms demand a significant degree of systems engineering activities. From this perspective, we state the following challenges focused in the SoS evolution:

Smart and runtime adaptation and reconfiguration: SoS with adaptive and autonomous behavior should provide mechanisms to change their behavior at runtime with minimal or none human intervention. Reconfiguration at runtime can be possible if the solution provided is feasible and performed in many cases under strict timing requirements;

Dynamic selection of choices: Many autonomous systems need to optimally make runtime decisions. Hence, variability management at runtime is still a challenging task for unforeseen scenarios;

Runtime management: The possibility of many critical complex systems to change from one operational mode to another demands mechanisms where the system variants can change at post-deployment time;

Awareness of runtime decisions: Software designers should be aware of which runtime changes can affect to critical parts of the architecture. Consequently, there is a strong need to notify the relevant stakeholders about critical runtime changes; and

Ripple effect of runtime decisions: As safety and reliability are important design concerns, critical decisions made by an autonomous behavior may impact on other critical parts of a complex system. Hence, we need solutions to track on the possible deviations of the normal system's operational mode when a change may cause damage on other parts of the system.

4. The Role of Dynamic Variability

Software product lines have proven as a successful technique for building families of related systems, often with a high degree of complexity. The evolution of a product line is achieved on behalf of software variability techniques [Capilla 2013], which maximize reuse and manage efficiently the variations of their family members. However, when complex systems demand runtime solutions able to manage the variations of systems dynamically, runtime variability becomes a relevant technique. Because managing the variations at runtime is still in the infancy, the emerging parading of Dynamic Software Product Lines (DSPLs) [Hinchey 2012] aims to manage the variations of systems at execution time. In this regard we discuss in this section those DSPL techniques that can be suitable for development and evolution of SoS systems that exploit context-awareness and runtime behavior.

Context-aware properties are the base to exploit runtime adaptation in many complex systems. The evolution of SoS can be achieved better modeling the contextual information using variability techniques. This contextual information of SoS that must be managed at runtime enhances the behavior of the system and supports better the evolution for unforeseen scenarios, as new functionality could be added, removed, or changed dynamically. In addition, system dynamics is an approach of systems engineering aimed to understand the behavior of complex systems over time. However, one step beyond on simulation of the behavior of complex systems refers to those solutions that can anticipate system changes at runtime. From a previous work we stated the challenge for managing runtime variability [Capilla 2011] as a promising solution to evolve SoS and which techniques can be suitable to model, change, and optimize the variations that happen when the behavior changes [Capilla 2014]. Consequently, runtime variability proves as a suitable technique for those SoS that demand adaptive and sometimes unpredictable behavior during execution time. In next section we discuss potential solutions at the architectural level for SoS that use context information and demand runtime adaptation in order to address the evolution challenges described in Section 3.

5. Airport Management Systems

Regarding the research methodology, in this preliminary work, we did an informal analysis of the target application domain based on the long experience of one of the co-authors. We did not performed an exploratory case study [Robson, 2002; Runeson 2009] or more formal analysis, as in this position paper we only wanted to state the main key areas where SoS development can be enhanced with dynamic variability techniques for the development of complex critical systems. In this light, one important domain suitable for SoS is the case of Airport Management Systems (AMS), which encompasses the automation of airport procedures in various areas, such as information systems, control and computerized remote controls, computer equipment, billing, baggage handling, security, weather information, controls signaling, and allocation of airport facilities, among others. All these key areas must work and cooperate in a coordinate and synchronized manner to handle the normal airport's operations and reduce human intervention. At present, the trend is to achieve maximum integration to maximize coordination and usage of resources and integrate middleware from different vendors in order to guarantee real-time operations. Consequently, reliability, safety, and security are major quality concerns addressed by this kind of system, many of them redundant in the AMS. Some of the subsystems that belong to AMS are the following:

Airfield lighting control systems: this system controls the lighting aids installed on the airport (runways, taxiways, stop bars, etc.), and in other cases light towers and obstacles. In small airports this system is often integrated with the power control system;

Weather information system: it is the responsible for acquiring, processing, presenting, recording, and disseminating information on the prevailing weather conditions at the airport and is vital for normal airport's operations. This system computes parameters extracted from a variety of sensors, such as wind speed and direction, atmospheric pressure, rainfall and humidity detection, and it can compute complex measures such as the Runway Visual Range (RVR);

Automatic baggage handling system: this system results key for the operation of large airports as it depends of the number of passengers and the season where passengers travel. It encompasses other subsystems that perform different functions with the luggage (e.g., billing, transportation, security, classification, etc.) and is considered one of the most cost-consuming systems in large airports; and

Airport security system: the aim of this system is to manage a wide variety of subsystems and services to control intrusion detection and the inner perimeter using CCTV cameras, access control, and fire detection sensors. Various security groups are associated to different roles and stakeholders.

Other systems belonging to the AMS are: the **electrical control system** to ensure an uninterrupted power supply using a large variety of sensors and analyzers, the **allocation of airport resources system,** which uses large databases and real-time information to allocate resources, and the **standard communication system,** which interconnects basic services and all voice, radio, and internet communications. Many other airport facilities can be integrated under the AMS, such as tunnel control system, passenger information system, slot management system, parking control system, airport GIS, aircraft docking systems, and many more.

As the large variety of subsystems and parameters is sometimes unmanageable for the many situations that may occur in an airport, the integration and configuration of all these system is hard. In many situations, the diversity and amount of the data managed by these subsystems as well as the responses and operations they need to perform depend on the state of other systems (e.g., a fire detection system can generate automatic actions on the access control system and the automatic baggage handling system, activating different alarms outside the airport). Consequently, the variations and the diversity of runtime scenarios complicate the maintenance and evolution when new requirements demand changes in the AMS. According to the SoS evolution challenges described in Section 3, and in order to address the large number of scenarios that may arise in the airport's daily operations, we identified the following challenges that AMS subsystems need to address regarding the dynamic adaptation and reconfiguration operations:

Challenge 1: Diversity of information sources where much of the information comes from the environment. As sensors belonging to different subsystems determine the airport configuration in real time, the AMS should provide a way to integrate all the information and distribute it to the implied stakeholders;

Challenge 2: Growing number of mobile employees that often use location services. The AMS should manage such diversity of information, aiming to exchange important data among systems and users in raw or cooked format, even outside the airport;

Challenge 3: Dynamic reconfiguration of systems that must be adapted to unforeseen situations. The huge number of interfacing systems and the kind of data they share becomes a problem to solve in case a system becomes off-line, in faulty situations or in maintenance mode, in most cases without interruption of the airport operation; and

Challenge 4: Multilingual and multicultural support. Airlines systems from different nationalities and different cultures must interact with the airport systems, so, the AMS should interact with all the airlines systems operating on it. In every season, those companies may change.

6. Building and Evolving AMS with Dynamic Variability

Considering the inherent characteristics and complexity of AMS and the challenges stated in the previous section, we identified the following opportunities where static and dynamic variability can play a role for building and maintaining some of the AMS subsystems, such as we describe below:

Variability in Airfield lighting control systems: multivendor solutions integrated in a single control system, include lighting control and single lamp fault signaling managed in real-time;

Variability in Weather information system: small airports do not need to calculate the parameters like Runway Visual Range (RVR) that might be replaced by a Visibility Measurement (VM). Other parameters like wind conditions or temperature are mandatory in big airports;

Variability in Automatic baggage handling system: the implementation of this system varies from a really simple distribution system, to complex systems with kilometers of installations and interfaces with many other systems, like fire detection or security systems; and

Variability in Airport security system: the security system may vary from a simple CCTV with a simple intrusion detection system in littlest airports, to really complex systems in wider airports, integrating thousands of cameras, thousands of sensors of any kind, and interfacing with many other systems in the airport or external to the airport.

6.1 Context-variability for AMS

Because many of the variations of these subsystems depend on sensor that analyze context information at real-time, we adopt one of the strategy to model AMS context properties using context variability techniques. As the large number of subsystems may complicate to model all the variability at the same time, we preferred to adopt a strategy focused on reuse, where context features are modeled using separated feature sub-trees and in the case a subsystem or part of it is replaced by a more modern one, the context variability sub-tree of that subsystem can be easily replaced in the feature model.

In Figure 1, we describe a layered reference architecture for AMS where we describe the organization of the subsystems mentioned and where dynamic variability can be used to managed the context properties of all these systems. Our approach uses a context variability model to describe both context and non context features (right side of Figure 1) but because the variability model is large, we only represent a subset of it. We adopted the strategy to have different branches in the feature model to discriminate between context and non-context features because of the following reasons: (i) non context features in AMS are more stable; (ii) a separate context feature sub-tree is more reusable in case we need to replace one of the subsystems; and (iii) context features that can modify the structural variability at runtime are easily to anchor in the feature tree under the right subsystem in case we need to add or remove features dynamically. By contrary, having two separate feature sub-trees for each subsystem may add more dependencies between features of each different subsystem.

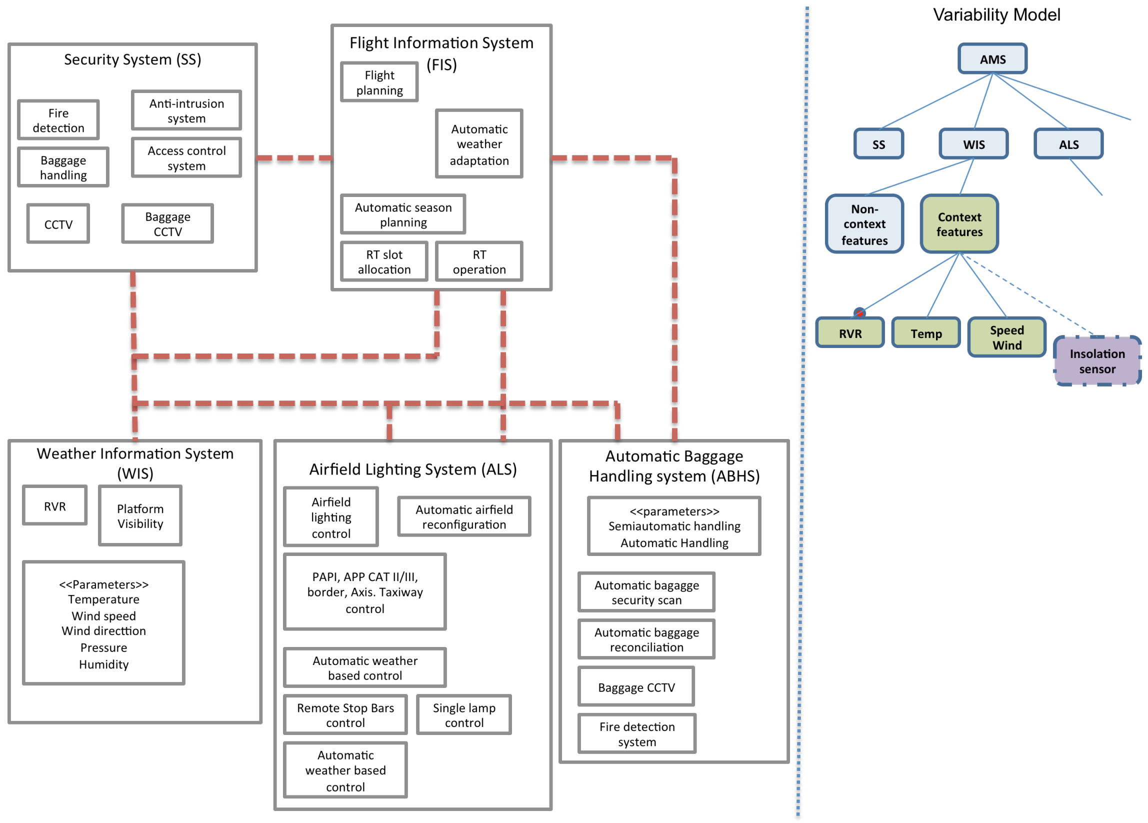
Security System (SS)
Fire detection
Anti-intrusion system
Baggage handling
Access control system
CCTV
Baggage CCTV
Flight Information System (FIS)
Flight planning
Automatic weather adaptation
Automatic season planning
RT slot allocation
RT operation
Weather Information System (WIS)
RVR
Platform Visibility
<<Parameters>> Temperature Wind speed Wind directtion Pressure Humidity
Airfield Lighting System (ALS)
Airfield lighting control
Automatic airfield reconfiguration
PAPI, APP CAT II/III, border, Axis. Taxiway control
Automatic weather based control
Remote Stop Bars control
Single lamp control
Automatic weather based control
Automatic Baggage Handling system (ABHS)
<<parameters>> Semiautomatic handling Automatic Handling
Automatic bagagge security scan
Automatic baggage reconciliation
Baggage CCTV
Fire detection system
Variability Model
AMS
SS
WIS
ALS
Non-context features
Context features
RVR
Temp
Speed Wind
Insolation sensor

Figure 1: Excerpt of the reference architecture of an Airport Management System (left side) and a subset of the context variability to model (right side) that describes context and non-context features and those that modify the structural variability at runtime (dotted lines).

6.2 Dynamic evolution of context variability

AMS have strong real-time requirements and runtime reconfiguration needs that require a different treatment from the evolution point of view. For instance, new features in the weather information system could be added at runtime through specific middleware. This could be the case that an AMS will need to incorporate an insolation sensor aimed to measure the amount of solar radiation energy received on a runaway. New context features could be added or removed dynamically in the feature model, having an clear impact on the structural variability of the AMS. Consequently, the evolution of the variability of these systems and subsystems can be managed using dynamic variability techniques such as those suggested in a previous work [Bosch 2012], and using the notion of super-types (i.e., a taxonomy to group features under a common functionality) to add and remove features at runtime. These dynamic variability techniques in combination with context features can reduce human intervention during critical AMS maintenance, as certain subsystems can be plugged into the AMS middleware at execution time.

7. Conclusions

The inherent complexity of SoS like AMS, which are composed by a wide range of systems and subsystems that must run coordinately at runtime, complicates the deployment and evolution of such systems. In this scenario, context variability and dynamic variability techniques, such as proposed in this work, become suitable for dealing better with the evolution of unforeseen situations and for modeling the variability and the diversity of scenarios that SoS must deal with. In this work, we have presented challenges and suggested opportunities for applying dynamic variability in the construction and maintenance of AMS. For the future work, we intend to get more evidence about the viability of applying dynamic variability to more efficiently control the construction of reconfigurable SoS, in the case, an AMS, and evolution of critical operations of such system.

References

Bosch, J., Capilla, R. (2012) Dynamic Variability in Software-Intensive Embedded System Families. IEEE Computer 45(10): 28-35.

Capilla, R., Bosch, J., Trinidad, P., Ruiz-Cortés, A., Hinchey, M. (2014) An Overview of Dynamic Software Product Line Architectures and Techniques: Observations from Research and Industry, Journal of Systems and Software 91(5), 3-23.

Capilla, R., Bosch, J., Kang, K-C. (2013) Systems and Software Variability Management, Concepts, Tools and Experiences, Springer.

Capilla, R., Bosch, J. (2011) The Promise and Challenge of Runtime Variability, IEEE Software 44(12), 93-95.

DoD. (2008) System Engineering Guide for Systems of Systems. Office of the Deputy Under Secretary of Defense for Acquisition and Technology, Systems and Software Engineering, Version 1.0.

Hinchey, M., Park, S., Schmid, K. (2012) Building Dynamic Software Product Lines, IEEE Computer 45(10), 22-26.

Maier, M. W. (1998). Architecting principles for systems-of-systems. Systems Engineering, 1, 4, 267-284.

Nakagawa, E. Y., et al. (2013) The State of the Art and Future Perspectives in Systems of Systems Software Architectures, In SESoS'13, Montpellier, France, 13-20.

Northrop, L., et al. (2006) Ultra-Large-Scale Systems: The Software Challenge of the Feature, Software Engineering Institute, Carnegie Mellon, Pittsburgh, USA.

Robson,C. (2002) Real World Research. Blackwell (2nd Ed.)

Runeson, P., Höst, M. (2009) Guidelines for conducting and reporting case study research in software engineering, Empirical Software Engineering Journal 14(2), 131-164

Sage, A. P., Armstrong, J. E. (2000) Introduction to Systems Engineering, Wiley Series.