Organizational Configurations in Executable Enterprise Architecture Models

Marie Ludwig¹, Nicolas Farjot¹, Jean-Philippe Babau², Joël Champeau²

¹ Thales Communications, 148 boulevard de Valmy
92704 Colombes Cedex, France
{marie.ludwig, nicolas.farcet}@fr.thalesgroup.com
² LISyC, UBO, UEB, 20 avenue Le Gorgeu
29200 Brest, France
jean-philippe.babau@univ-brest.fr
³ LISyC, ENSTA Bretagne, 2 rue François Verny
29200 Brest, France
joel.champeau@ensta-bretagne.fr

Abstract. Architecture models provide a legible description of the system, and help describing its properties in a representation shared and understood by most stakeholders. “Executable” models allow running enterprise behaviors in a convincing way to support the understanding of complex enterprises, such as Systems of Systems (SoS). In the context of an approach to enterprise architecture evaluation through model execution, we have designed a modelling framework and tools based on Domain Specific Languages (DSL) to enable rapid prototyping and iterative analysis of architecture models. This paper presents our experience in the integration into this DSL of support for defining different structural configurations for a system, and performing reconfiguration at runtime.

Keywords: Systems of Systems, architecture modeling, enterprise configuration.

1 Introduction

The presented work addresses the scope of Systems of Systems (SoS) architecting and engineering. In particular, we focus on architecture analysis and evaluation, especially in a dynamical context.

SoS have been characterized by Mark W. Maier in [1] by five main features:
- Operational Independence of the Elements – Each system composing the SoS can separately operate;
- Managerial Independence of the Elements – Systems have an independent design and acquisition process;
- Evolutionary Development – The SoS is dynamically formed, systems are added or removed;
- Emergent Behaviour – The capabilities of the SoS are more than the sum of capabilities of the systems which compose it;
- **Geographic Distribution** – The geographic extent means systems interactions are based on information and should rely on a Network.

Because the addressed engineering challenges and solutions are also valid for large systems and more generally all kinds of networks of cooperating entities, in the remaining of this paper, we will use ‘enterprise’ as a generic term comprising both large systems and Systems of Systems.

Section 2 introduces the background of the project and the state of the art. Section 3 introduces the challenge of enterprise entity configuration. Section 4 presents the metamodel elements we identified to define entity configurations. Sections 5 and 6 conclude with future works and perspectives.

### 2 Background

Enterprise architecting and engineering raises multiple complexity issues. Especially in military and homeland security contexts, enterprises have to adapt to unpredictable environments and evolving missions, enable dynamic collaborations, master critical service chains, and mitigate vulnerabilities.

Model-Based Systems Engineering (MBSE) approaches, which are currently mainly used in IT enterprises, emphasize the use of an architecture model to provide a legible description of the studied system shared and understood by all stakeholders. The model may also be used and possibly refined at all stages of the lifecycle of the enterprise engineering as a common reference. Structured knowledge about the enterprise, organized in a model, is an enabler for the identification and understanding of the key aspects of its architecture, despite an increasing complexity in organization and service chains. In particular, the model provides support when dealing with the following architecting challenges:

- Identifying the key operational capabilities of the enterprise and mastering their availability;
- Identifying and mastering the critical service chains across the enterprise;
- Identifying and mastering the key resource flows.

All the stakeholders (operational analysts, architects…) may contribute to the refinement of the model from their own point of view. This allows them to build and share a common vision on the enterprise architecture and provides a common reference to tackle the different concerns.

#### 2.1 Executable Architecture Models

In case of an executable architecture model, short design-experimentation loops allow an incremental understanding to the enterprise complexity and a rapid prototyping approach to evaluate its architecture.

*Executable* model can of course mean many different things. Model animation, e.g. based on UML state charts or sequence diagrams, is a kind of execution that provides an illustration of some dynamic aspects of a model. Simulation tools such as process execution tools help understanding business behaviors within an architecture model. However, current process execution approaches drawn from the BPM (Business
Process Management) community seem to miss some key aspects of enterprises executions such as command and control and collaborative organizations, services and service-level agreements, roles and responsibility, etc. We think that a more holistic approach to enterprise modelling and simulation is required, e.g. addressing service and organizational aspects and allowing multi-scale and multi-viewpoint analyses. This approach is based on the execution of enterprise architecture models.

**Benefits.** Executing a model does not only provide a better understanding of the architecture, but is also a support for architecture variants comparison and evaluation. The architecture models allow conducting objective and quantitative analysis of the variants they represent, in order to:

- Check the appropriateness of the resources (human resources, systems, capabilities...) to the needs of the enterprise missions and estimate the potential weak points of the architecture:
  - Identify probable overloaded systems to plan alternatives for the systems development;
  - Identify probable roles that could lead to overloaded operators, to redefine them or redistribute their activities;
  - Identify probable bottlenecks in the processes or communication channels;
- Conduct coarse-grained measures of performance and effectiveness along service chains to have objective criteria for the comparison between variants.

**State of the Art.** Architecture Frameworks such as the DoDAF [3] and the NAF [2] allow the creation of architecture models according to different views and diagrams. These formalisms were initially designed for system acquisition stakeholders. In their current version, they provide an extensible coverage of architecture matter in a rather complex fashion that fails to enable the global simulation of the dynamic aspects of the enterprise being modeled. Their meta-model has been designed for communication between architects and procurement agencies rather than for model execution.

Several communities have envisioned approaches and developed tools to address the challenge of architecture models simulation. In [4], Pawlowski et al. from the MITRE Corp. present a methodology to “import key products of the DoD Architecture Framework into an executable form”. Their approach is based on the transformation of DoDAF models designed with System Architect tool toward simulation tools. These simulation tools leverage simulation metamodels. Their ICAMS (Integrated C4ISR Analysis and Management System) middleware supports the creation of simulation data from an architecture model, as well as the mapping between the simulation elements and the architecture elements.

The Defence R&D Canada has published a survey and assessment study of current Model-Based Systems Engineering (MBSE) standards with respect to their support for executable architectures. They identify executable architectures, i.e. “the dynamic model of the behavior of a system where the architecture elements are uniquely identified, consistently used and structured in a way to enable their simulation” [5], as a key asset for linking architecture models and simulation. Their report recommends the use of a combination of multiple modelling standards deriving from MDA.
(Model-Driven Architecture), among others UPDM (UML Profile for DoDAF and MoDAF) and xUML (Executable UML), and HLA (High-Level Architecture) for simulation interoperability.

HLA is a general purpose architecture for distributed simulation systems, that is defined under IEEE standard 1516 since 2000 [6]. It addresses interoperability across a federation of simulations, but focuses on the technical infrastructure and lacks support for solving conceptual level problematic (global architecture validation, verification…). The MDA, using UML as the modelling standard, has also been envisioned as a candidate for supporting a homogeneous conceptual layer for a HLA federation [7]. Kewley et al. presented a systems engineering approach to federated simulation development using MDA [8], that aims to ensure conceptual interoperability for all the simulation models.

In our case, we seek the ability to enable round-trips between architecture design and architecture simulation in a seamless fashion, with no model transformation or information loss involved. Our aim is to run enterprise behavior in a convincing way for the various stakeholders conducting and participating to the capability engineering. We have a need for a model that can be executed as a whole, i.e. by taking into account all its aspects. In particular, we execute operational processes in correlation with enterprise operational entities, services, roles, and command and control organizations.

2.2 IDEA Methodology and Tools

To support enterprise architecture design and evaluation based on rapid prototyping and iterative analysis of architecture models, we have designed a modelling framework and execution tools, which aim to work on the exact same model. This approach based on the execution of semi-formal models is part of a process named IDEA bridging wider CD&E (Concept Development & Experimentation) and system engineering through architecting.

This process is composed of four stages:
- Identify: Identification and analysis of the needs and problem space,
- Design: Candidate architectures construction,
- Experiment: Validation, evaluation and optimization of the candidate architectures,
- Assess: Acceptance of the selected architectures.

The IDEA methodology aims at providing support for a close collaboration between the various stakeholders (operational analysts, domain experts, system architects…), especially regarding the problem space and needs analysis, the definition of evaluation criteria and the identification of candidate architectures. This methodology is more thoroughly presented in [9].
Because the proposed tools are to be use for rapid prototyping by stakeholders that may be operational experts with limited abilities in modelling, we had to go for domain-specific languages (DSL) instead of general purpose languages such as UML. We designed graphical notations and chose underlying concepts to be easily understood by the participants of CD&E sessions. The core concepts of the IDEA metamodel are close or similar to the ones described in the Architecture Frameworks such as the NATO Architecture Framework (NAF) [13]. They are meant to describe what the enterprise is supposed to do (e.g. capabilities), how it can achieve its goals (e.g. collaborations) and who performs the tasks (e.g. humans or systems). A synthetic view of the core concepts and their relationships is presented in Fig.1.

3 Enterprise Entities Configuration

In the IDEA metamodel, Enterprise Entities represent the actors who achieve the tasks the enterprise is intended to realize. In particular, the systems composing a System of System are seen as Enterprise Entities. Enterprise Entities can be physical (such as vehicles) or logical (such as groups like patrols). The structure of the Enterprise Entities organization is based on several kinds of decomposition relationships, whose properties, behaviors and operational meanings are different depending on the kind of entity being decomposed. For the sake of simplicity, in this paper all these kinds of relationships will be equally referred to using the generic term Enterprise Entity decomposition relationship.

Dynamicity and Organizational Flexibility. Dynamicity and flexibility can be considered as fundamental properties of an enterprise organization, in particular for

Fig. 1. IDEA Enterprise Architecture metamodel core concepts and relationships.
Systems of Systems composed of mobile platforms. The organization of an enterprise may vary even when composed of an identified set of systems. This organization may change during an operation due to events or new orientation of the mission. Dynamicity is present in different aspects of the enterprise lifecycle: *when the enterprise is set up* in an incremental way, connecting the systems then assembling the capabilities of systems; *during the planning* where a platform can be replaced by another one with identical or similar capabilities; and *during deployment and operations* where platforms may be added or removed due to an occurring event.

From the previous observations, we can draw that there is a need for managing organization configurations during the whole enterprise lifecycle. In particular, from the early stages of architecting, enterprise architecture models such as IDEA models shall support the definition of Entities configurations, *i.e.* different structures and organizations for a same Entity. At runtime, model execution shall enable reconfiguration, *i.e.* the modification of a given entity structure or organization that may be triggered by a given operational situation.

**Use Case Presentation.** The examples used in the following of this paper will be based on a simplified use case close to one of our current industrial projects in the domain of public safety. We consider a *Multi-Purpose Police Car* that embeds two equipments racks which may contain different kinds of devices depending on the mission the car is assigned to. Both racks can contain a *Vehicle Tracking System*, the *Front Rack* can also contain a *Mobile Data Terminal* and the *Back Rack* a *Speed Recognition Device* as illustrated in Fig.2.

![Fig.2. Police Car equipment options example presentation.](image-url)
4 Approach and Implementation

Modelling different configurations for a same entity raises close challenges to features and variability modelling [10]. Variability modelling implies describing the features of a product and guiding the selection of features allowing the construction of a specific product – which is similar to the selection of a given set of children for a specific configuration of an Enterprise Entity [11].

The Abstract Entity Concept. We draw inspiration from the Variation Point notion of Variability modelling, which materializes a location where variation may occur, to introduce in the IDEA Metamodel a concept named Abstract Entity that materializes a location in the Enterprise Entity organization where different configuration options are available.

For a given Abstract Entity, the configuration options are restrained to an identified set of Entities referred to as its Variants. These Variants can be Enterprise Entities or other Abstract Entities. Abstract Entities can be targeted by an Enterprise Entity decomposition relationship. Fig.3 summarizes the way the Abstract Entity concept has been introduced in the IDEA Metamodel. Fig.4 illustrates how the equipments options of the Multi-Purpose Car can be modeled using two Abstract Entities, whose variants are the devices that can be contained in the racks.

Fig.3. The Abstract Entity concept in the IDEA Metamodel.

Configuring an Abstract Entity means choosing one of the Variants. In the case of the Police Car, it would mean choosing one kind of equipment for each rack.

At runtime, performing reconfiguration on an Abstract Entity means substituting another Variant to the one that has been previously chosen. In the model, this involves the redirection of all relevant links it had to other model elements.
Entity Configurations. Any Enterprise Entity whose decomposition relationships lead to an Abstract Entity is considered as Configurable, which means it has the possibility of having different configurations. We formalized this notion under the following conditions:

- An Enterprise Entity is Configurable iff…
  - It is the source of a decomposition relationship that targets an Abstract Entity
  - OR
  - It is the source of a decomposition relationship that targets an Entity that is Configurable.

The notion of configuration of a configurable Enterprise Entity depends on the configuration of its children. The impact each child may have on the configuration of its parents depends on its type:

- If a child is an Abstract Entity, defining a configuration for its parent implies choosing one Variant for this child;
- If a child is a configurable Enterprise Entity, defining a configuration for its parent implies choosing one configuration for this child;
- Trivially, if a child is a non-configurable Enterprise Entity, it has no impact on the configuration of its parent.

Fig. 4. presents the Configuration concept we introduced in the IDEA Metamodel.
To prevent a model from containing incoherent patterns, the relationships showed on Fig. 5 are constrained by a set of rules. In particular:

- If an Enterprise Entity is not configurable, it cannot have Configurations.
- The child configurations and the child variants of a Configuration must belong to child entities of its Enterprise Entity.
- A Configuration cannot contain two child configurations (resp. child variants) that belong to the same Enterprise Entity (resp. Abstract Entity).

During model execution, performing a reconfiguration implies modifying the decomposition links of the entities in the model to switch from one configuration to another. The reconfiguration may be constrained by the availability of the resources required to build the new configuration.

**Application to the Use Case.** Let’s consider two different configurations for the Multi-Purpose Police Car:

- In *Patrol Configuration*, the front rack of the car contains a Mobile Data Terminal and the back rack a Vehicle Tracking System.
- In *Road Surveillance Configuration*, the front rack of the car contains a Vehicle Tracking System and the back rack a Speed Recognition Device.
Fig. 6. details the model of the two configurations as created using the concepts presented in Fig. 5. (For legibility reasons, the entity decomposition relationships between the car and the racks are not represented).

Fig. 6. Detailed Configuration definition of the Police Car.
5 Perspectives

In an enterprise, flexibility and configuration challenges are not limited to the Enterprise Entity organization. In particular, we plan to introduce configuration aspects for the operational roles as well. Operational roles interact with each other by consuming and producing services and exchanging information. The support for the reconfiguration of these interactions is a major enabler for the constitution of dynamic collaborations of entities with a common mission interest and information needs.

Because of the organizational flexibility brought by the configurations, the entity organization at runtime is likely to evolve and the emergence of new capabilities could come along with undesired effects. Thus there is a need for functionalities and processes that retain and maximize operational capabilities through the control and optimization of resources, services and interactions in conjunction with the evolution of the operational situation. Such functionalities are usually referred to as System Management. In the ongoing of the project, we also plan to integrate the support for System Management in IDEA models at runtime.

6 Conclusion

We designed a metamodel and a tool suite to support rapid design – execution prototyping loops of an architecture model. The model is built according to three main points of view, capabilities (what), enterprise organization (who), and the roles that are used to map capabilities to enterprise entities (how). Flexibility in the enterprise organization is introduced by the Abstract Entity concept, which materializes an alternative in the entity decomposition. Different Configurations for an Enterprise Entity can be modeled depending on configurations of children enterprise Entities and variant choices for the children Abstract Entities. We successfully used these metamodel concepts to describe different vehicle configurations in the domain of public safety.

7 References

3. NATO Architecture Framework (NAF), http://www.nhqcs.nato.int/