15th ICCRTS - 2010
"The Evolution of C2"

Evaluating Enterprise Architectures through Executable Models

Topics – Modeling and Simulation | C2 Architectures and Technologies

Marie Ludwig, Nicolas Farcet

THALES LAND & JOINT SYSTEMS
Battlespace Transformation Center
160 Boulevard de Valmy
92704 Colombes CEDEX FRANCE
+33(0)1 46 13 32 67
marie.ludwig@fr.thalesgroup.com

Abstract

This paper presents an approach to evaluate and compare enterprise architectures through the execution of semi-formal models. To support this approach, we have designed a modeling framework and tools based on Domain Specific Languages (DSL) to enable rapid prototyping and iterative analysis of architecture models.

"Executable" models allow running enterprise behaviors in a convincing way to support the understanding of complex enterprises, such as Systems of Systems (SoS). These models enable to describe the structure and organization of operational capabilities such as C2, with a focus on service-based interactions and collaborative processes between the enterprise entities.

Defining responsibility domains for the entities in the model allows partitioning the information flows, and helps to disseminate the relevant information to the proper actors. In a SoS, communication interactions not only follow the hierarchical command chain, but also happen across this chain, within consistent collaborations of entities or Communities of Interest (COI). The model supports allocation of roles, reconfiguration of service relationships inside a COI, and degraded modes analysis.

Execution of enterprise architecture models provides an efficient way to assess their consistency and refine them through gaming (table-top, role-playing…). Judiciously chosen execution measures of performance enable to compare architecture variants with respect to business metrics.

Introduction and objectives

As the complexity of the modern large enterprises or System of Systems (SoS) increases, new system engineering and architecture challenges emerge to help the stakeholders capturing the whole dimension and identifying the key aspects of the enterprise[1]. During the early stages of architecting, there is a need for a gradual understanding of the structure and confidence in the ability of an enterprise to fulfill its missions and satisfy its objectives. 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, which helps describing its characteristics and provides a common representation of the system shared and understood by all stakeholders.

Architecture Frameworks such as the DoDAF or NAF 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.

[1] In the remaining of the paper, we will use 'enterprise' as a generic term comprising systems of systems.

In our case, we intend to use architecture modeling as a support for architecture analysis and decision and engineering governance rather than as a sole communication and filing means. Thus we have a need for architecture prototyping and evaluation tools and methods.

State of the art

Several communities have envisioned approaches and developed tools to address the problematic of architecture models simulation.

In [3], 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. Ring et al. also define an "*integrated, unambiguous, and consistent*" architecture as the prerequisite to any type of analysis purpose; *integrated* meaning based of a suitable set of DoDAF products and defined by a conceptual Architecture Specification Model (ASM) [5].

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*" [2], as a key asset for linking architecture models and simulation. Their report recommends the use of a combination of multiple modeling 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 modeling 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.

Our approach aims to provide support for system architects as well as business experts for the understanding of enterprise architecture in the early stages of architecting, and enable short design/execution loops and rapid prototyping. In this case, there is a need for a directly executable architecture model, that does not require model transformation to be simulated. Compared to typical waterfall model-based approaches to engineering leveraging code generation, in our approach the model execution is likely to be used from the start of the architecture development cycle to enable seamless and incremental design-execution loops.

Background

In this paper, we present an approach to enterprise architecture design and evaluation based on the execution of semi-formal models, which 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:

- **I**dentify: Identification and analysis of the needs and problem space
- **D**esign: Candidate architectures construction
- **E**xperiment: Evaluation and optimization of the candidate architectures
- **A**ssess: Acceptance of the selected architectures

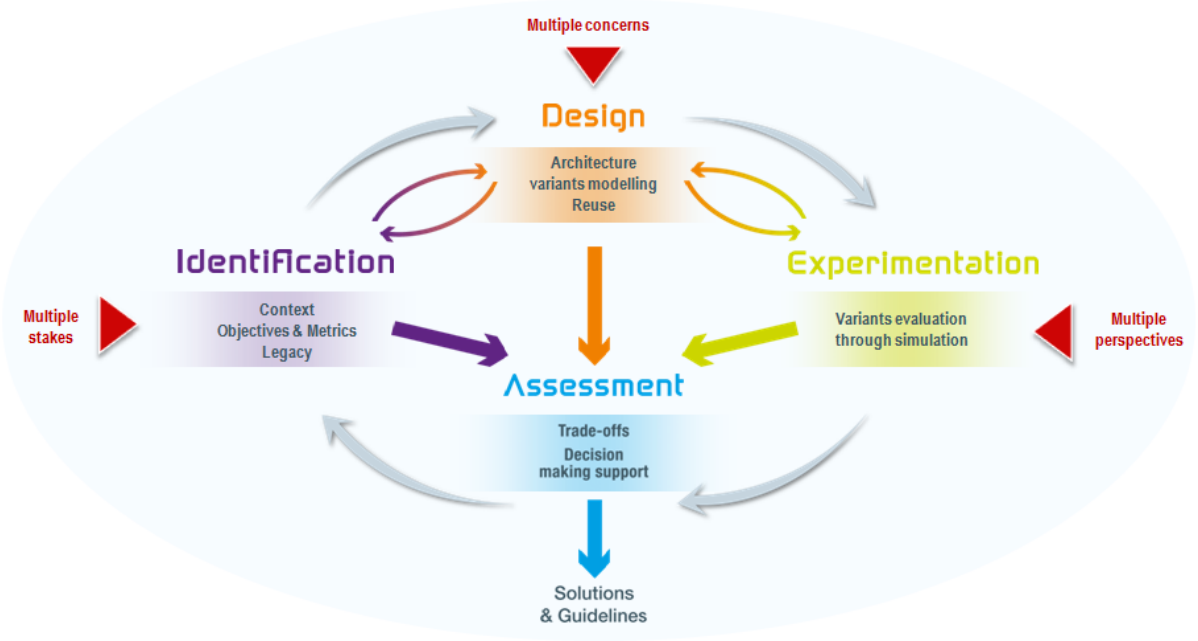

Figure 1 - IDEA process overview.

The main principle of this process is the modeling and execution of architecture models created from a capability analysis. The IDEA methodology enables 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.

One cannot assume that capability engineering is a top-down or a bottom-up analysis process. The support methodology shall thus be flexible with respect to this process. Our approach for tooling the IDEA process was to leverage our core architectural models through a set of modelling perspectives called *capability map, logical architecture,* and *enterprise architecture*. These modelling perspectives are adopted when adding more information to the model, but they do not impose a specific analysis order. For example, one can start to analyse and model the legacy processes and organizations, or one can start to analyse and model the target capabilities of the enterprise. The tools must support top-down approaches (based on existing capabilities to be allocated on new organizations) as well as

bottom-up (modelling legacy organization and their supported capability). This tooling flexibility was mandatory to adapt to a large variety of methodological contexts (per stakeholder, per viewpoint).

Once initial architecture models are built as outputs of the "D" step, they can be reworked offline and then given back to participants ("D-I" loop). Model-centric approaches enable to capture this material in a formal way, which guide and structure the further reflections. Executable models allow interactive debugging, as well as a collaborative evaluation of the relevance of the architecture towards the business needs and the ideas that emerged during the creativity meetings.

"Executable" model: the key to rapid architecture prototyping

"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 model execution is required, e.g. addressing service and organizational aspects and allowing multi-scale and multi-viewpoint analyses.

We seek the ability to run enterprise behaviors in a convincing way for the various stakeholders conducting and participating to the capability engineering. Our aim is to create 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. Our metamodel was designed with execution in mind. The goal is to enable round-trips between architecture design and architecture simulation in a seamless fashion, with no model transformation or information loss involved. Most of the tools of our suite thus have the ability to exchange models without transformation, even though they address different concerns (modeling, validation, evaluation…). Each tool contributes to fleshing out the architecture model with concerns corresponding to specific points of view, for specific stakeholders, at specific times in the engineering process.

Overview of our formalism and tools

To support this approach of rapid prototyping and iterative analysis of architecture models, we have designed a modeling framework and tools, which aim to work on the exact same model. Because the proposed tools are to be use for rapid prototyping by stakeholders that may be operational experts with limited abilities in modeling, 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. Because enterprises can be complex in terms of goals, organizations, and behaviors, it is important not to restrict expressiveness of architecture models. For our modeling foundation, we have designed a core set of architectural concepts close or similar to the ones described in the Architecture Frameworks. The concepts of the set have been chosen for their simplicity and expressiveness – e.g. *capability, service, product flow,*

process, role, enterprise entity, and *command chain* –, and organized into a semi-formal modeling framework. A simplified view of the core concepts and their relationships is described in Figure 2 below.

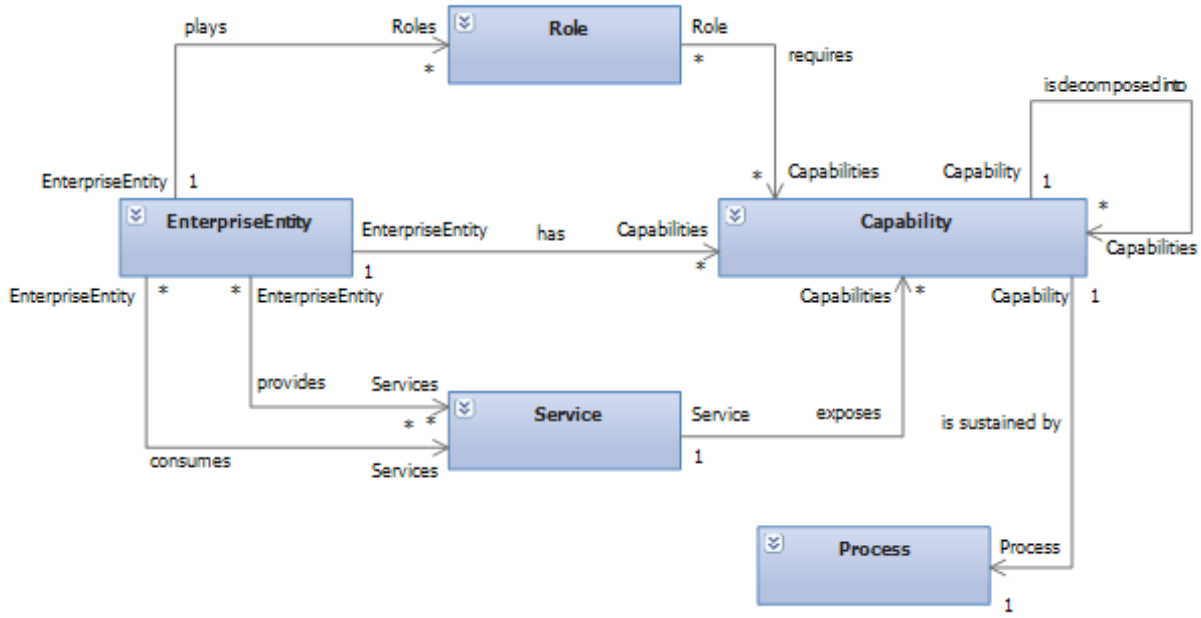

Figure 2 - IDEA Metamodel core concepts overview (simplified)

Our suite is composed of two main tools, IDEA Designer and IDEA Performer.

IDEA Designer is a modeling tool, and proposes a multi-viewpoints approach to the creation of the architecture model. This approach differs from the one adopted in the current Architecture Frameworks, in the fact that it decomposes the enterprise according to the different perspectives it addresses rather than using layers such as *Operational*, *System* and *Technical*. According to the IEEE-1471 Standard [1], a viewpoint is defined by "*a way of looking at a system. A viewpoint captures the conventions for constructing, interpreting and analyzing a particular kind of view*". Each viewpoint is composed of views that enable the modeling of specific aspects of the enterprise relative to a set of architectural concerns. In the tool, each view can be built using one or more associated diagrams that provide dedicated toolboxes and menus.

IDEA Designer also includes a model execution module, named Analyzer. This module allows debugging the architecture models and visualizing their dynamic aspects, by running the processes and the interactions across the enterprise. The processes are executed in parallel, may be synchronized and take into account their organizational context.

IDEA Performer is the model performance evaluation tool. It leverages the same model execution mechanisms as the Analyzer and allows conducting dynamic performance measurements and analysis of the models. The measures produced by this evaluation are meant to support architecting choices and decisions. Evaluation metrics can be freely added to models, and are defined by the user in accordance with their concerns.

Use case: fusion of observation information

In the remaining of the document, we will illustrate the presented approach and framework with a simple use case that leverages the problematic of Information Fusion. As part of a larger enterprise, we have several sensors that gather observation information for the node that embeds them. The node has to correlate the information received from its sensors, and send the correlated information to a hierarchical superior node. Superior nodes gather all the information from the nodes that belong to their domain of responsibility and correlate them to obtain a whole picture of the situation. The aim of the study is to compare several patterns for the information flows, for example to assess the relevance and efficiency of cross-hierarchical chain information flows.

Modeling and model validation

In our modeling approach we separate artifacts related to enterprise capabilities (i.e. **what** the enterprise is able to do) from those related to enterprise organization (i.e. **who** does it). This separation is a key asset for envisioning different organizations fulfilling the same objectives or different behaviors for the same organization.

Capability modeling

The *capability* concept is at the core of our formalism. It allows for a simple grasp of an Enterprise's abilities through a multi-scale and modular description of its complexity. In particular, the *capability map* of an enterprise presents a consistent overview of its capabilities, processes, and services at multiple levels of granularity. The tool has been created for multi-scale modeling, which means it does not impose the granularity level of the model. It allows describing high-level macroscopic operational capabilities as well as technical elementary capabilities of sensors or effectors, and thus lets the user choose the most relevant level with respect to the needs of their study.

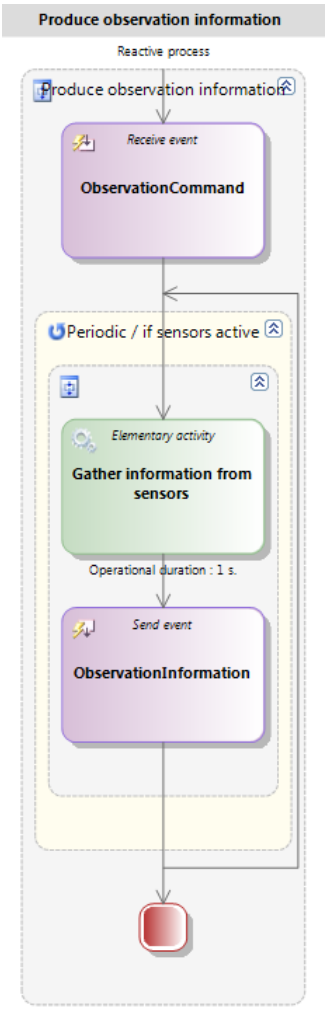

Figure 3 - Example process : Produce observation information

- In the case of our information fusion system, a trivial capability map contains only *Start sensors*, *Produce observation information* and *Perform information fusion*.

- The processes are expressed using a workflow formalism. We modeled the process leveraged by the *Produce observation information* capability as a simple sequence of three activities, *Receive observation command, Gather information from sensors* and *Send observation information*. The two latter are executed periodically. Since it was not relevant toward our information flows study to detail the inner functioning of the sensor gathered in the *Gather information from sensors* activity,

we kept it very coarse-grained. If we later have to discuss the efficiency of specific classes of sensors, we will refine this behavior to take sensor performances into account.

Enterprise entities modeling

The enterprise organization can be described in terms of operational and organic command chains, which represents the hierarchical relationships between the elements, and composition, i.e. the members of an organizational group (ex. members of a platoon) or the equipments on a platform. The enterprise organization diagrams also enable the creation of Communities of Interest (CoI). CoIs do not necessarily depend on any hierarchical interactions and provide support for cross-hierarchy information sharing: typically, the members of a same CoI share a certain amount of information and exchange various services.

- For example, the Fusion System use case elementary organization has been described as two platforms equipped with each two sensors (composition relationships : brown links with diamond ends in Figure 4 below), under hierarchical responsibility of a third platform (hierarchical relationships : blue “OpCommand” links).
    - Each of the two subaltern platforms is responsible for the observation of a specific area, but needs the information that results of the correlation of the observation results of both areas.
    - The leader does not perform any observation, but needs the correlated observation information as well.

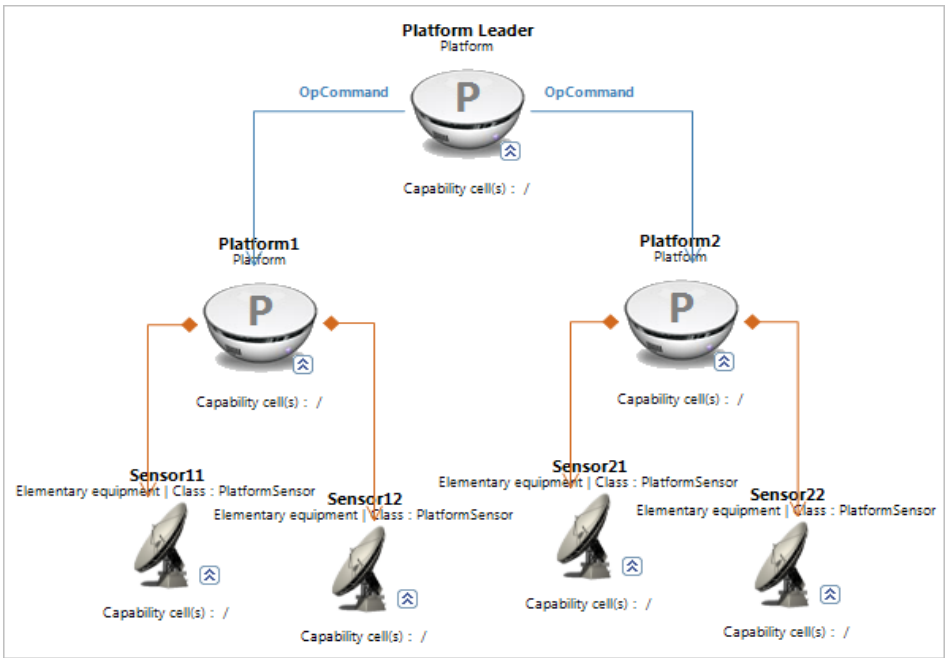

Figure 4 - Sample Enterprise organization (IDEA Designer screenshot).

For legibility purposes, the sensors won’t appear on the following figures.

- We will consider two different ways of routing the information flows, which has a strong influence on how CoIs will be created in this organization. For example:

- **Variant 1:** the information flows along the Command Chain, and Platform1 and Platform2 do not communicate with each other. They both send their information to the leader, who performs the fusion and broadcasts the global information. The information flows are represented by strong blue arrows in Figure 5 below. In this case, the model contains two CoIs, one containing Platform1 and Leader, the other containing Platform2 and Leader, as illustrated on Figure 5 below.

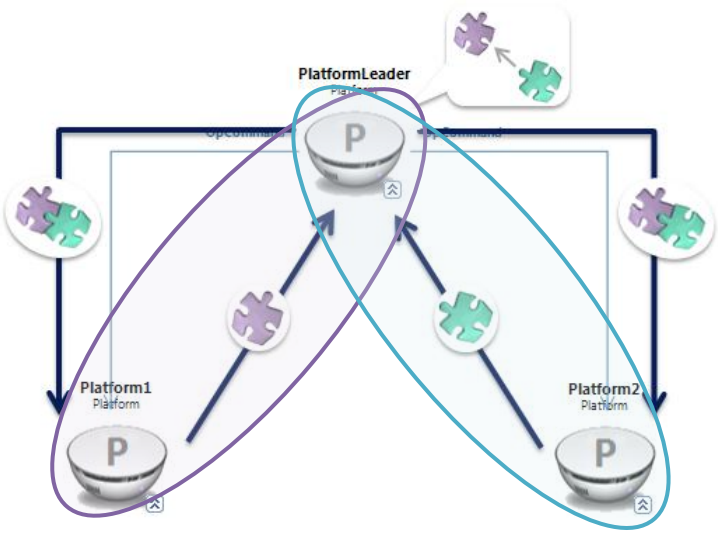

Figure 5 - Information flows example variant 1: along the command chain

- **Variant 2:** Platform1 and Platform2 directly share their observation information. There are information flows along the hierarchical chain as well as across it, and the model contains one CoI that includes all three platforms, as illustrated on Figure 6. On contrary to the first variant, there is no more circulation of the global information since each of the platforms performs a local fusion.

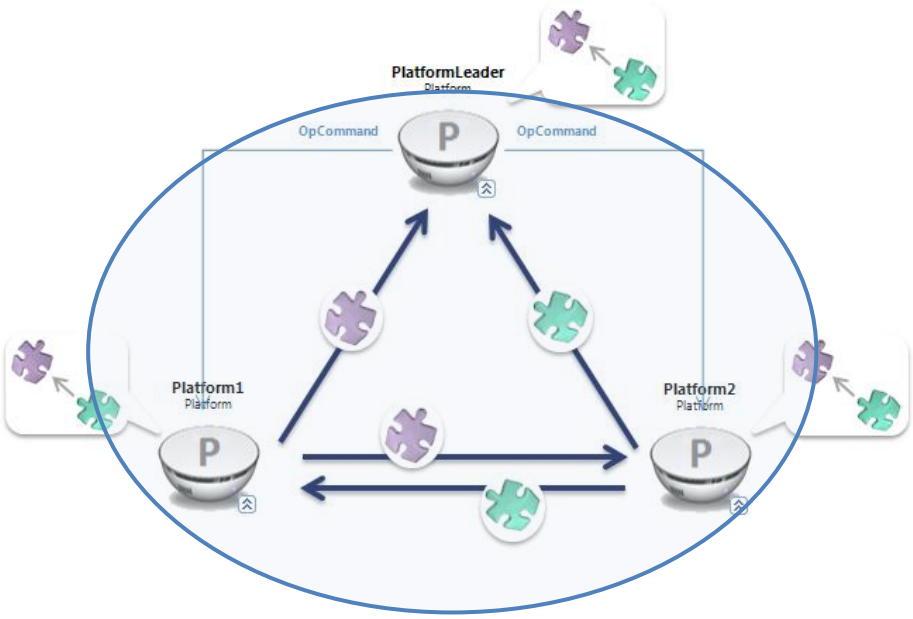

Figure 6 - Information flows example variant 2: along and across the command chain

Roles and interactions modeling

The mapping of the capabilities on the organization is enabled by the design of operational roles. These roles can be considered as the cornerstones of the logical architecture of the enterprise, and gather the key operational behaviors of the enterprise in terms of capabilities, service interactions and information flows. The interactions between the "physical" enterprise entities are kept consistent with the logical interactions described between the roles. The tool supports two communication interactions paradigms, service-oriented and event-driven.

As an example, let's consider the design of roles for the three platforms of the draft Information Fusion System Variant 1:

- Two roles are identified: *Observation module* (with capabilities *Start sensors* and *Produce observation information*) and *Fusion module* (with capability *Perform information fusion*).
- An *Observation module* sends its observation information to the *Fusion module*, and waits for the fusion node to send back the fused information. This interaction has been designed as an exchange of events.

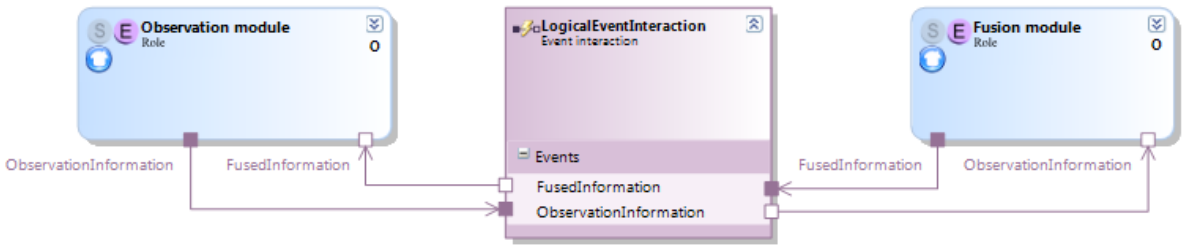

Figure 7 - Message exchanges between roles (IDEA Designer screenshot)

- The mapping of these roles on the platforms is rather intuitive, Platform1 and Platform2 being *Observation modules* while Platform Leader is a *Fusion module*.

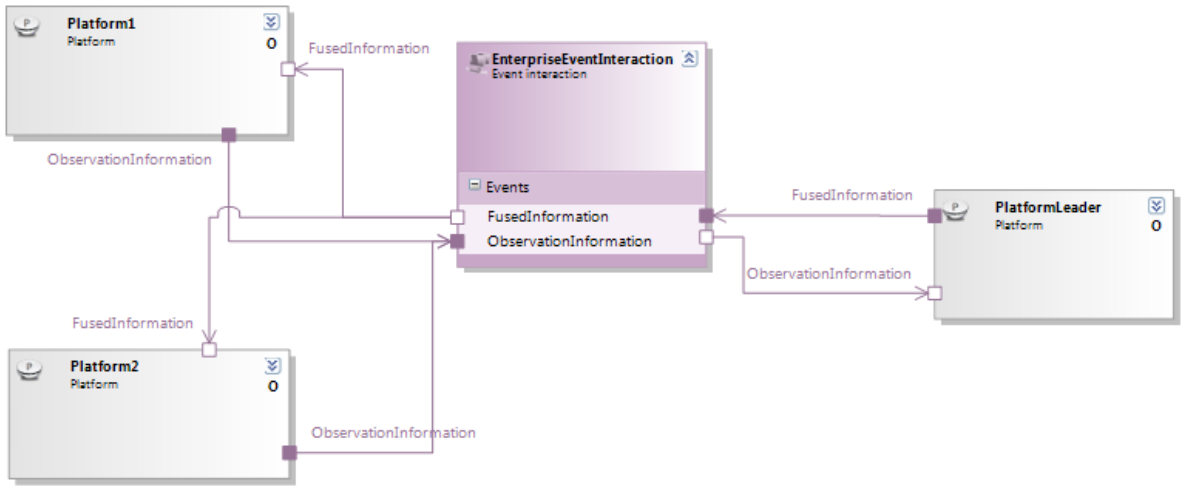

Figure 8 - Message exchanges between platforms (IDEA Designer screenshot)

Model debugging

Enterprise models are likely to be large and complex, with intricate interactions between the various entities. Executing the model helps understanding the model, and thus allows checking that it meets the expectations of the various stakeholders. This enables to achieve convergence between the operational analysts' and the architects' visions. Our approach also recommends executing the model at all stages of its creation, not only when it has been completely built. Short and seamless execution / model consolidation loops enable an incremental debugging and domain relevance checking of the model and thus help managing its complexity.

The Analyzer execution module included in IDEA Designer offers three main views on the model during its execution:

- A view of the capabilities of the enterprise entities, whatever their granularity. This view allows starting/stopping capabilities during the simulation, and checking their execution status.

- A view of the step by step execution of each process, that enable to focus on the internal behavior of each entity and supports discussion about the relevance of the way it has been modeled. The user chooses which processes they want to display.

- A logical deployment view of the entities and the messages they exchange, that enable to focus on the interactions between the entities and supports discussion about the way the information flows have been routed. Layers representing the hierarchical chain or the CoIs can also been displayed if the user needs them.

Architecture variants execution and evaluation

Executing a model does not only help 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...) to the needs of the enterprise missions, and identify 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 performance and measures of effectiveness to have objective criteria for the comparison between variants.

Measures of Effectiveness (MOEs) *focus on the system's capabilities to achieve mission success within the total operational environment*. MOEs represent the customer and user views. Measures of

Performance (MOPs) *characterize physical or functional attributes relating to the system operation, measured or estimated under specified testing and/or operational environment conditions* [10]. MOPs represent the architect and engineer views.

Measures of Performance and functional rules

Such quantitative analyses can be performed even if the model has been done with a coarse granularity. In this case, their results will not give precise information about the characteristics of the enterprise, but will allow identifying key elements of the architecture, such as vulnerabilities, critical service chains, degraded modes, etc.

IDEA Performer simulation tool runs measures of performance and operational rules during an execution of a model. *Measures of Performance* usually represent functional or non-functional metrics that are considered important with respect to the architects' needs. For instance, measuring the traffic on the various communication channels is a key point for the evaluation of our Information Fusion system. *Functional rules* represent either operational, system, or technical behaviors that may be discriminatory for the architecture variant. In our Information Fusion use case, the fusion algorithm (centralized or distributed…) is as important as the platform organization for the efficiency of the whole enterprise.

Although IDEA Performer tool does not impose any kind of format, the results of the measures are usually stored in a database and presented through traditional Excel diagrams similar to the one presented in Figure 9 below. The parameters of the measures are often based on the value of a specific set of enterprise entity properties, defined in the model and updated during its execution.

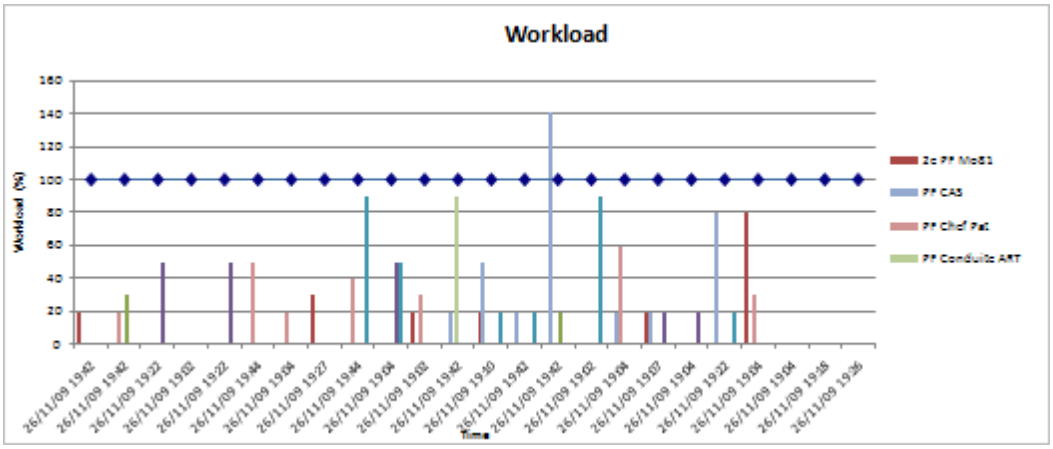

Figure 9 - Measure example: Entities workload. (x : operational dates, y : workload, bar colors represent the concerned entity as listed in the legend)

Advanced evaluation

Real enterprises, especially in a military context, have to face continuously evolving situations and must be able to reconfigure their structure or their behavior to maximize their ability to accomplish their missions. The execution of the architecture model shall not be limited to a visualization of the

dynamic aspects of the architecture, but shall also support dynamicity in the architecture itself (represented by the IDEA model). Reorganization and reconfiguration of the modeled architecture at runtime support the evaluation of dynamic organizational doctrines and guidelines. In particular, service and event interactions can be reconfigured during simulation, either by the simulation operator, or by operational rules.

As an advanced use of the functional rules framework within IDEA Performer and the Analyzer module of IDEA Designer, it is possible to define reconfiguration rules that will be triggered under certain conditions during the execution.

- As a simple illustration, let's consider an elementary Information structure similar to the one which is described in Figure 6. The three platforms of the group share their observation information and belong to a single CoI (Variant 2). During the mission, there is a need for integrating a third observation platform (Platform3). In the model file, the reconfiguration rules add Platform3 as a participant of COI, and then give it access to the communication channel allocated to the group.

- At the end of the execution, the tool enables to save the model file, which allows re-opening it with IDEA Designer to study thoroughly the modifications that have been made during the simulation.

Technology considerations

From a software point of view, our tool suite has been developed using the Microsoft .NET framework. The meta-model that describes the structure of our architecture models has been designed with the DSL Tools add-in for Visual Studio. IDEA Designer leverages Visual Studio shell version. Both the Analyzer module of IDEA Designer and IDEA Performer leverage a runtime environment for the execution of processes and activities, services, events, interactions, product flows, and external behaviors. This environment is based on a set of .NET technologies: Windows Workflow Foundation (WWF), Windows Communications Foundation (WCF), Windows Presentation Foundation (WPF), and Bing Map for the cartographic aspects.

In particular, we extensively leveraged a technology brought by DSL Tools, namely consistency and validation rules. To be able to execute the model in a convincing way, all its aspects (processes, organizations…) must be kept coherent even if they have been defined in different modeling viewpoints and diagrams. The consistency between and within the various viewpoints is ensured in our framework by an integrated verification environment, which constantly guides the user and ensures the executability of the model:

- **Consistency rules** forbid the user to create incoherent architecture patterns that would prevent the model from being executable. For instance, the tool does not allow a same equipment entity to be embedded on two different platforms. Consistency rules and related undoing is enforced by a transactional engine. The current engine is however not thread-safe which requires some care if adopting a multi-client design.

- **Validation rules** are launched when the model is saved or on-demand by the user, and raise errors when the model is incomplete. For example, the Information Fusion model won't be validated if there are Platforms playing the *Fusion module* role but no one being an *Observation module*: it would mean that some entities are waiting for information that no other entity can provide. The user can ignore the validation errors, for instance to postpone the design of some parts if it needs further discussion.

Both tools of our tool suite implement the whole set of consistency and validation rules. This ensures that at any time the model is consistent, and prevents them from trying to execute a non-validated model.

Discussions and ongoing challenges

This tool suite has already been used on some concrete use cases. These use cases validated most of our design choices and provided useful feedback both from the design, and the usability point of views.

The metamodel we designed to support our architecture evaluation approach has a core consisting in about 150 concepts, 100 relations, 200 consistency and validation rules, and about 20 diagrams. Although these numbers are not impressive per se, the overall design of the IDEA tool suite raised a number of challenges.

Firstly, maintaining executability was a constant challenge. Each new addition to the metamodel (concept, relation, or rule) required a deep thinking about run-time aspects to ensure that design choices would not break the overall executability. This is a reason why we chose from the start to rely on a single core metamodel. Initial attempts have been made to leverage parts of the UML2 metamodel as building blocks of our metamodel (in particular, the class, property, and instance related aspects). However, runtime management complexity and lack of native executability have refrained us from going in this direction. We built upon a simple set of concepts (mainly capability, process, role, interaction, enterprise entity). Instead of grounding these concepts on a class-instance framework such as in UML structures, we adopted an approach drawing from prototype-based languages [9]. Indeed, prototype-based languages provide a simpler approach to manage copy and derivation of model-elements at runtime.

In a first version of our metamodel, we had duplicates of some concepts (namely capabilities in the capability map, and capabilities required by roles). We had to maintain traceability relationships across diagrams and it implied multiple and complex consistency rules (to cope for the adding, deleting, renaming, etc. of capabilities). In a more recent version of our metamodel, we got rid of this duplication and gained robustness and simplicity.

One of the challenges of our modeling tool was to expose architectural concerns so that the architects would not be constrained in their thinking. As it has already been mentioned before in this paper, one particular decision we took was to decorrelate capability aspects (capabilities, services, products, processes) from enterprise organization aspects (enterprise entities and their organizational relationships). Then the role diagrams allowed the architect to elaborate multiple architectural variants built by combining capabilities patterns into organizations schemes.

Ergonomics consideration in the Designer revealed the need for a copy-paste mechanism. Such a feature may sound elementary, but its implementation raised a certain amount of questions related to scope and granularity of architectural constructs. Choosing which elements should the copied, which should be propagated, and how to reconnect others to existing elements in the model has been a real design challenge.

To summarize, here are key recommendations we ended up when designing our language for executable architectures (this is just good sense, mostly):

- Think about executability right from the start, and not afterwards. Rendering an existing metamodel executable is close to impossible in our opinion.
- Think simple and minimal (progressively elaborate metamodel and execution language). Of course making things simple takes time (we have refactored our metamodel multiple times since many years).
- Define modeling viewpoints and associated diagrams with respect to what power and flexibility you want the architect to end-up with (in our case we wanted to easily recombine architectural assets to bring up architecture variants in terms of capabilities vs. organizations). Unfortunately, some existing modeling tools have multiple diagrams exposed for exhaustivity reasons mostly. In our future version, we will leverage WPF technology for diagrams enabled by Visual Studio 2010 to explore new usability areas.
- Finally, ensure that the modeling tool can be used in multiple methodological contexts with multiple backgrounds in terms of legacy to take into account in the architecting process. Indeed, the architecture can be captured either starting from capabilities, processes, services, organizations, or roles, depending on what needs to be clarified first or depending on who is available to provide the knowledge first. This is why we had in mind from the start the requirement to elaborate architecture incrementally by capturing knowledge from experts according to multiple viewpoints. This was possible thanks to our approach to capture architecture constructs: each new information gets aggregated to the architecture through the application of consistency and validation rules.

Perspectives

Validation through Tabletop gaming and Role playing

IDEA Performer was designed to include a module which enables multi-team interactive gaming, for business processes and human factors collaborative evaluation. This feature is however still under implementation. The tool has been designed to support both tabletop gaming and role playing. Tabletop gaming and role playing are facilities for collaborative realistic exercises for a group of stakeholders (e.g. system architects, operational analysts…), by providing them with a common vision of a chosen situation and environment for the SoS.

- Tabletop gaming enables a collective analysis of the behavior of the enterprise by giving the participants the opportunity to act on the operational situation and react to subsequently raised events.
- Role playing enables to concentrate on a subset of behaviors in the enterprise, by focusing on specific roles and proposing situations that enables a refined analysis of the different aspects of these roles.

When the studied enterprise has to include legacy processes, interactive gaming can also be used to collect information on an existing doctrine, in a more natural manner for the domain experts than through knowledge capture tools or white boards. In this case, the way the human roleplayer acts and reacts to help capturing and modeling business activities.

Extension of the simulation scope

To extend the simulation opportunities and offer a more realistic vision of the model at runtime, we have thought of coupling our IDEA Performer with existing simulation tools. The goal of this perspective is to take advantage of environmental models and detailed operational rules that have already been implemented such as: terrain, mobility, deployment, platform, and equipment models. Our approach would be to interconnect our simulation engine with other simulation engines through the exchange of timed-events.

For example, it may be interesting for a more sophisticated version of our Fusion model to take into account the performances of the sensors on the platform (terrain, error rates…). During the execution of such a model, the Player runs the first activity of the Observe process and asks a sensor simulation tool to scan the area and calculate the detection using some properties of the sensor element in the model (location, range…). Based on this information, fusion activities implemented in IDEA model might be drawn.

Improving interoperability with Architecture Frameworks

Architecture Frameworks such as DoDAF and NAF support the execution of some aspects of an architecture (eg : animation of activity diagrams), they were not designed to support executability of an architecture as a whole. They nevertheless define architecture concepts shared by more and more actors. We have added to IDEA Designer a set of NAF views that are generated from our metamodel. We think these views can be interesting when entering system engineering activities. We leverage IDEA to support architecture evaluation and tradeoff activities, then produce NAF views for the selected architecture candidates. We plan in the future to extend our NAF coverage and interoperate with other NAF tools.

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*). Executing an architecture model at various stages of its conception allows debugging the model, as well as verifying its conformity with respect to the operational experts' vision. Measures of performance and functional rules provide objective and comparable support means for the evaluation of variants of an architecture. Advanced evaluation features such as architecture model modification at runtime enable to experiment reconfiguration doctrines useful to study operational robustness issues. Ongoing works for our approach include enabling multi-team interactive gaming for model evaluation, and bridging our suite with alternative simulation tools. The approach and tools have already been experimented successfully on use cases.

Acknowledgements

We would like to thank Mr E.Biau for his contribution in the definition of the Performer tool, as well as Ms H.Bachatène, Mr A.Gauthier, Pr J-P Babau, Mr B.Aizier and Mr J.Champeau for their review and advice on this paper.

References

1. ANS/IEEE Standard 1471 Website, http://www.iso-architecture.org/ieee-1471/index.html

2. Bagnulo, A., Harvey, G., "Executable Architecture – Survey and Suitability Assessment Study of Systems Engineering Standards and Methodologies", Defence R&D Canada report, March, 2008.

3. Pawlowski, T., Barr, P., Ring, S.J, "Applying Executable Architectures to Support Dynamic Analysis of C2 Systems", 2004 Command and Control Research and Technology Symposium, 2004.

4. Gardiner, J., "Linking Architecture Models and Synthetic Environments – An Integrated Project Support environment", 7th Annual Conference on Systems Engineering Research, 2009.

5. Wisnosky, D., Vogel, J., Ring, S., "The Road to Executable Architectures", November 2004, http://www.wizdom.com/Presentations/roadToExecutableArch.ppt

6. IEEE Computer Society, "IEEE Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP)," Report 1516.3, 2003, http://ieeexplore.ieee.org/servlet/opac?punumber=8526

7. Tolk, A., Muguira, J., "M&S within the Model Driven Architecture", Interservice/Industry Training, Simulation and Education Conference (I/ITSEC), 2004.

8. Kewley, R. Jr., Tolk, A., "A Systems Engineering Process for Development of Federated Simulations", Spring Simulation Multiconference, 2009.

9. Ungar, D., Smith, R., "Self: The Power of Simplicity", OOPSLA '87 Conference Proceedings, pp. 227-241, 1987.

10. Roedler, G., Jones, C., "Technical Measurement – A Collaborative Project of PSM, INCOSE, and Industry", Technical Report, 2005.