Customization principles of an aeronautics SLM environment and an illustration on aeronautics use cases: the doors management system and the flight control system

Odile Laurent, Airbus; Ivo Viglietti, Alenia SIA; Fabien Paganelli, SAGEM; Stéphane Bonnet, CNRS; Gérard Cristau, Thales TRT; Nikolaos Priggouris, Hellenic Aerospace Industry; Philippe Baufreton, SAGEM

Abstract--- Having efficient means to build an appropriate system engineering framework is a differentiating factor for the competitiveness of the European aeronautics industry. We present in this paper an approach and a set of concepts enabling the construction and customization of a system engineering environment. These customization principles, set up in the frame of the CESAR project, are driven by a multi-view model-based system development process and by a system data description. The flexibility and the benefits of this approach are demonstrated on two industrial cases: the doors management system and the flight control system.

Index Terms*---CESAR, System engineering, system development framework, Customization, Interoperability*

General introduction

System engineering is a key subject in order to address the ever increasing complexity of systems in aeronautics industry. A significant effort must be achieved, at company level (aircraft manufacturer, tier 1 suppliers, etc) and at supply chain level to deploy adequate tools and information technologies with the aim to support fine tuned system development engineering practices. Indeed, the variety of systems, the diversity of company engineering practices, the vast number of tools (COTS, Open Source and in-house developed) available make very difficult the set up of a consistent System Life cycle Management environment (SLM), able to manage complex system life cycle data and processes.

Having an efficient aeronautical system information backbone to which the various design tools can be easily interfaced is a real challenge and a key enabling factor for Aeronautics European Industry competitiveness.

CESAR (Cost -Efficient methods and processes for SAfety Relevant embedded systems) proposes the concept of RTP (Reference Technology Platform) whose aim is to provide industry with a set of enablers that can support the construction of company dependant SLM. Therefore, the RTP is not a monolithic framework, but a set of tools and services supported by appropriate methodologies and processes, that offer significant place for customization according to the needs of application domains or even on a per organization basis.

The RTP customization principles developed in the frame of CESAR include: on the one hand, tool plug-in facilities to allow integration of existing, commercial or in-house tools, in the execution platform; and on the other hand, facilities to define in a top-down approach the selection of reference platform components aiming to support engineering activities. This takes into account the deployment constraints of the information technology infrastructures on which the SLM will be installed and executed. The benefit of such an approach is twofold: (i) it provides companies with flexibility offering the choice of engineering tools, services infrastructure and hardware platform while mutualizing common components, (ii) it allows software editors to develop alternative implementation of services infrastructure that can lead to innovations.

In this paper, section 1 will give a brief overview of the CESAR RTP, section 2 will depict the RTP customization mechanisms, section 3 will illustrate the RTP capabilities and customization mechanisms on concrete examples, an aircraft Doors Management System and a Flight Control System Monitor application. Finally the conclusion will draw some perspectives for aeronautics Engineering and SLM platforms.

1. A CESAR RTP overview

The Reference Technology Platform developed in the frame of the CESAR project is a set of tools delivering services, meta-models and connectivity means from which, the construction of a customized platform supporting specific system engineering needs can be achieved.

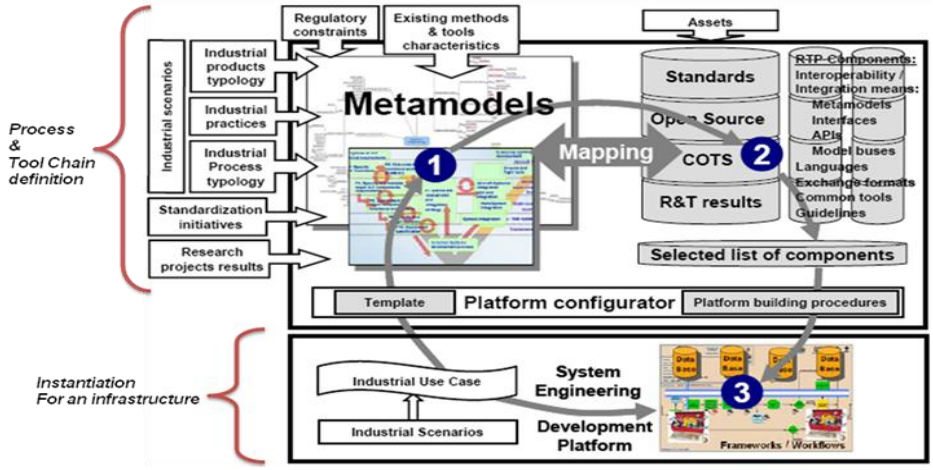


Figure 1 the RTP components

The above figure shows the CESAR reference technology platform components. These include meta models (1), tools and infrastructures features (2) that allow building a system engineering framework (3) that fits specific industrial needs. This customization process depends mainly on the engineering process and on the kind of the industrial application to be supported: the process workflow and the process steps generally drive the selection of the tools and their underlying meta-models (focus of section 2.a), while the industrial application fully guides the specification of the system engineering data model on which a concurrent multi-viewpoints1 modelling approach relies (focus of section 2.b). This system engineering data model can be considered as application dependant and should be distinguished from the other meta-models that are either languages specific (e.g. AADL [1]) or tool specific (e.g. FIACRE [2]).

2. Customization principles

The picture below describes the inputs and steps that are necessary to build a tools execution platform on which the engineering tools can be run to support the system development activities.

The RTP customization flow starts with the selection of the proper reference practices for system development, from the Practice Library. The latter is the input for assembling the desired development activities description. Detailed relevant information in the process description enables the configuration and deployment of an RTP instance tailored in terms of tools chain and tasks as described in the sub-section 2.a.

In order to support RTP operations and tools chain interoperability engineering, data-models are specified and the concept of RTPI (RTP Interface) is introduced. The RTP interface corresponds to the part of an RTP service (e.g. create an artifact version) that can be used by other services and applications. The exposed parts of the RTP services are standardized so that, the implementation of a service is completely decoupled from its usage.

The notion of system engineering data model, that ensures the consistency between the different models managed all along the development life cycle, is defined. (See 2.b)

[1] A specification of the conventions for constructing and using a view. A pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis [IEEE Std 1471-2000], where a view is a representation of a whole system from the perspective of a related set of concerns [IEEE Std 1471-2000].

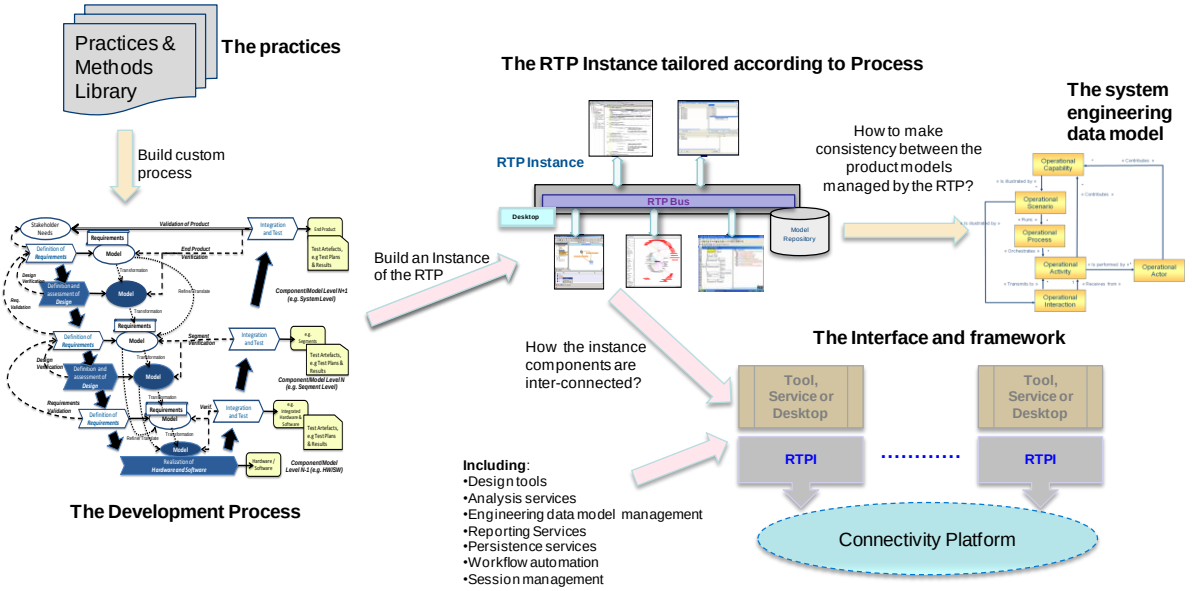


Figure 2 The RTP customization principles

a. Using the CESAR practices libraries (CPLs) for the RTP Customization

Difficulties met by a development team of complex systems when planning and then implementing multi-disciplinary projects requires a comprehensive support from project/process definition towards process automation.

A knowledge base including all the aspects to be represented and maintained in both planning and implementation phases is required.

The suggested approach is to completely model, through instantiation of re-usable patterns, the engineering process needed by the project being planned. To address this, the creation of a Practice Library is proposed with the purpose of gathering the definition of tasks, activities and any information relevant to the engineering process.

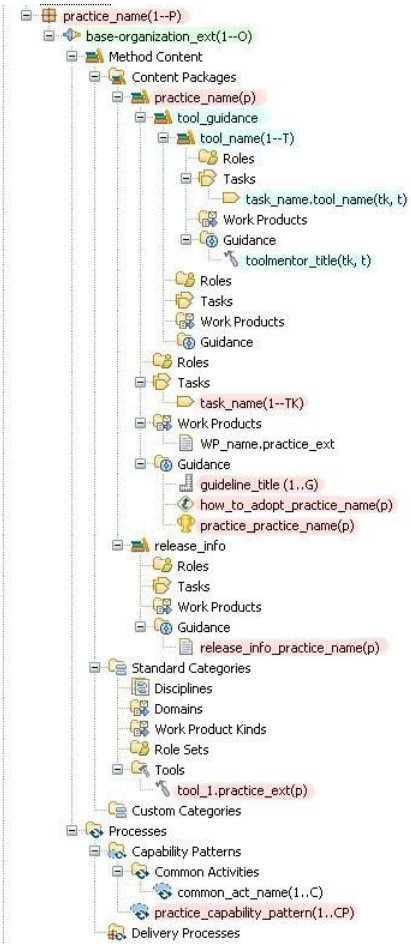
This Practice Library can be viewed as a knowledge base that provides:

Ready-to use elementary method contents including
- Common roles
- Common work products
- Tools information
- Common tasks

Structured method chunks, such as
- Capability patterns (i.e. reusable sub-processes)
- Complete reference processes (e.g. from standards)

Capability patterns are the main bricks used during the instantiation of a specific process. They can be considered as “prototypes” that can be replicated to create sub-processes instances (“process chunks”). The relations among the “prototype” and its instances can be set in such a way that modifications in the prototype are either automatically reflected to the instances or are fully independent which means that the instance may be completely modified in place.

See picture on the right for an illustration of the basic structure for a practice in the CPL.



The definition of tools that will be part of the tool-chain for the instantiated engineering process is also supported. Tools can also be mapped to specific tasks that are part of the instantiated engineering process.

The assembly and structuring of the process knowledge described can be materialized with the support of an authoring tool compliant with the formalism SPEM [3] selected by CESAR. This process description language is widely adopted in the process engineering domain and well supported by tools.

As mentioned earlier, the concept of RTP Interface has been introduced in order to support interoperability within the tool chain. It implies to specify the "reference" functions that can be offered and invoked by the components of the RTP instances.

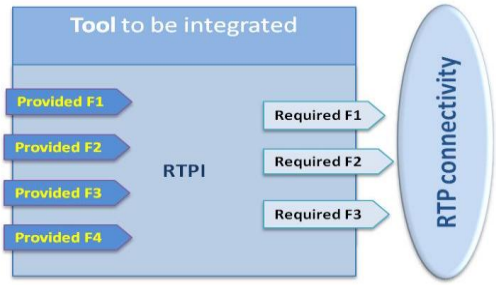


Figure 3 – Tool Integration

A formalized "RTP Instance configuration model" has to be generated starting from the customized information as described above. It shall include the enactable tasks derived from the process knowledge and the project description information, including registered users, involved resources and work products.

This conceptual flow of actions for customizing and deploying an RTP Instance is depicted in the picture below.

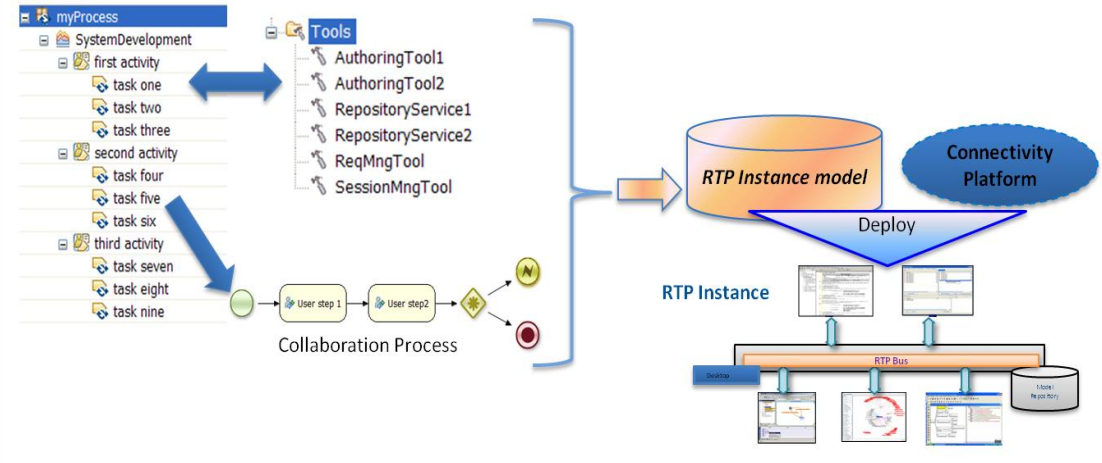


Figure 4 – RTP Instance Specification and deployment

Once an "Instance Model" has been defined it is still possible to complete or update it by means of a dedicated interface provided by the RTP.

This customisation and deployment approach, due to the formal and structured nature of the knowledge it handles, enables a progressive evolution path that leads to an almost complete automation of the tool execution platform bringing an optimisation in projects set-up.

b. Building a system engineering data model

In complex aeronautics systems, different specialists from various disciplines interact to build a common product in a concurrent engineering environment.

The different system specialists thus manage their activities in parallel implying, most of the time, a duplication of the same data. In such a context, ensuring data consistency all along the development life cycle is extremely difficult and may lead to discrepancies that are costly to solve.

Based on that, the essential purpose of a system engineering data model is to ease collaborative work on data which are shared among several teams and that are crucial for their disciplines. It is achieved by enabling unambiguous sharing of information and ensuring that data are properly updated when necessary.

A common system engineering data model is identified as the common part of the development data managed by the different disciplines through the development life cycle activities. The core system engineering data model corresponds to the data that are common to all the disciplines.

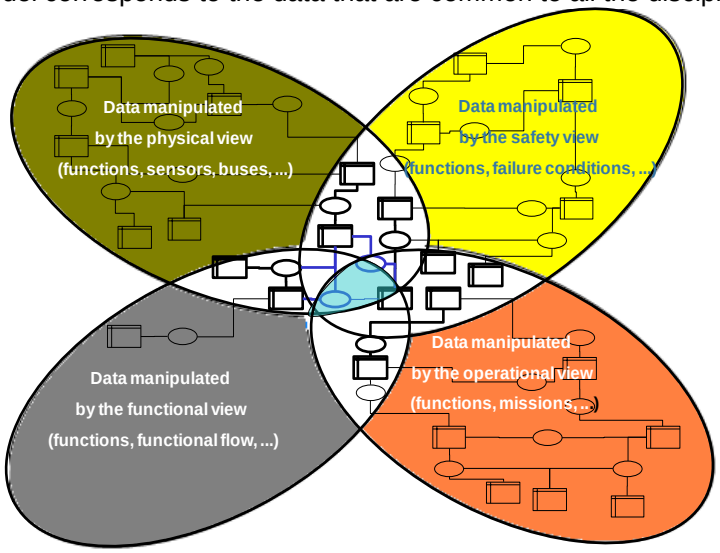


Figure 5 - An illustration of a system engineering data model

The figure above shows an example of a system engineering data model supporting four viewpoints (physical, safety, functional and operational). The common system engineering data model corresponds to the white sectors: data shared between at least two viewpoints. The core data model is highlighted in blue.

A system engineering data model, manipulating development data managed by different specialists relying on different processes, is dependant of the application domains and of the industrial processes. So, a generic approach must be set up to specify a system engineering data model. This approach will be customized by each company according to its organization and needs.

The different process steps needed to define a system engineering data model are shown on the following figure:

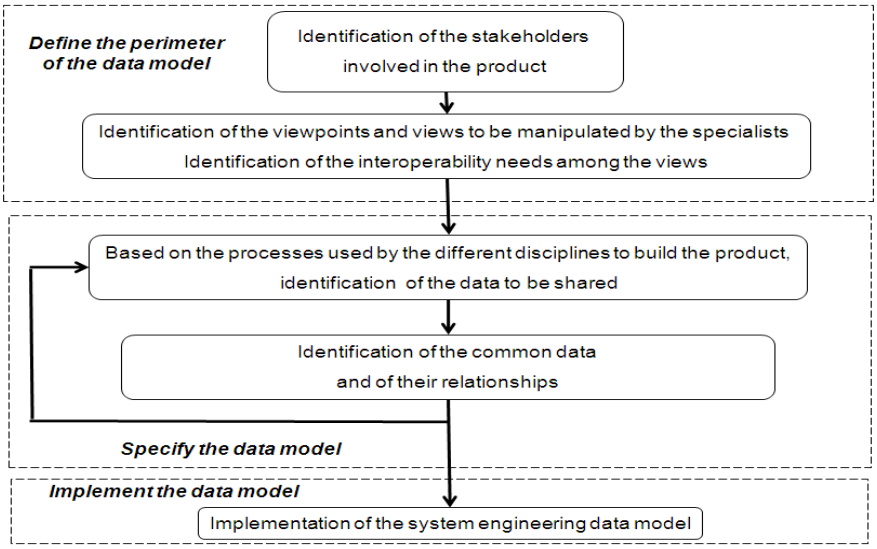


Figure 6 - The steps of a system engineering data model definition

The first step consists in identifying the different disciplines that have to work in tight cooperation. From this identification of disciplines, the views and the associated relevant models to manage in the model-based engineering approach are deduced. This phase allows pointing out all the actors that will have to work together.

The second step is crucial as it deals with the identification of the potential data to be shared. These potentially shared data are identified by analysing the processes used by each discipline to develop the product. In particular, the outputs of activities that are produced by one discipline and consumed by a set of activities led by other disciplines are potential candidates for sharing.

After having identified the potential data to be shared; only the core data used to build the foundations of the system design activities are selected as common shared data and will be part of the common system engineering data model. In order to succeed in this approach, all the stakeholders of the system development activities have to be involved.

This approach has been applied on the simplified doors management system as described in the section 3.a.

3. Customization principles applied to aeronautics use cases

In this section we proceed with illustrating the concepts elaborated above in 2 aeronautics related examples.

a. A system engineering data model for the Simplified Doors management system (SDMS)

i. Brief description of the Simplified Doors Management System (SDMS)

The SDMS is derived from the operational doors management system that is present on all the Airbus aircrafts. Its purpose is to manage the passenger and cargo doors of the aircraft. Since uncontrolled decompression at altitude is a catastrophic event, marks this system as safety-critical for the aircraft.

The following figure shows the doors managed by the A350 doors management system.

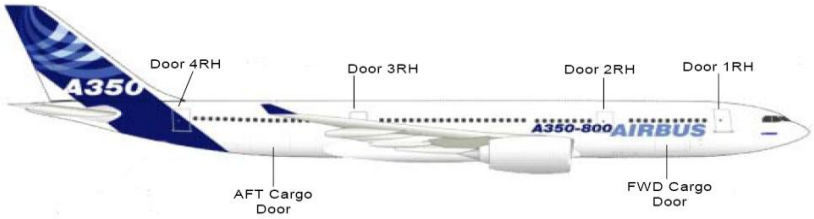


Figure 7 - The A350 doors managed by the SDMS

The SDMS ensures four main functions:

- Management of the indication of the aircraft external doors and the associated evacuation slides status on the flight deck,
- Inhibition of the pressurisation of the aircraft if at least one aircraft external doors is not closed, latched and locked,
- Control of the residual pressure warning lights for external doors,
- Control of the flight lock on the passenger doors in order to prevent deliberate door opening in flight.

ii. A SDMS data model for architecture design and exploration

In order to specify a data model useful for the SDMS in the architecture design and exploration phases, the different methodological steps defined in the section 2.b were followed.

1. The first step: define the perimeter of the data model

To keep the architecture design relatively simple, the architecture exploration process is only performed for the Pressurization prevention function of the SDMS, on two passenger doors, using only performance and safety-related criteria (end-to-end latencies between the door sensors and the

pressurization authorization signal, probabilities of occurrence of failure conditions identified during the functional hazard assessment, total weight of the physical architecture components).

The objective is to find an architecture solution for the system that satisfies the functional, safety, timing and weight requirements.

The architecture modelling was thus done considering the needs of three engineering teams: the safety team, the architects' team, the team responsible for managing the aircraft weight constraints.

These teams identified four viewpoints to address: the functional viewpoint, the physical viewpoint, the performance viewpoint and the safety viewpoint.

2. The second step: specify the data model

The data of the SDMS necessary to define the different viewpoints described above were analyzed. The SDMS sub-functions of the Pressurization prevention function (e.g. acquire doors sensor, authorize pressurisation) and the physical components (e.g. a remote data controller, a sensor) were considered as the key elements to be shared between the development teams. These functional data (called Function, FunctionPort in the data model described in figure 8) and physical data (called Implementation Component, ImplementPort in the data model described by the next figure) are qualified by a property concept that depends on the viewpoint. For instance, failure conditions related to the safety viewpoint are associated to functions, while hardware component failure probabilities are associated to implementation components as shown on the figure below.

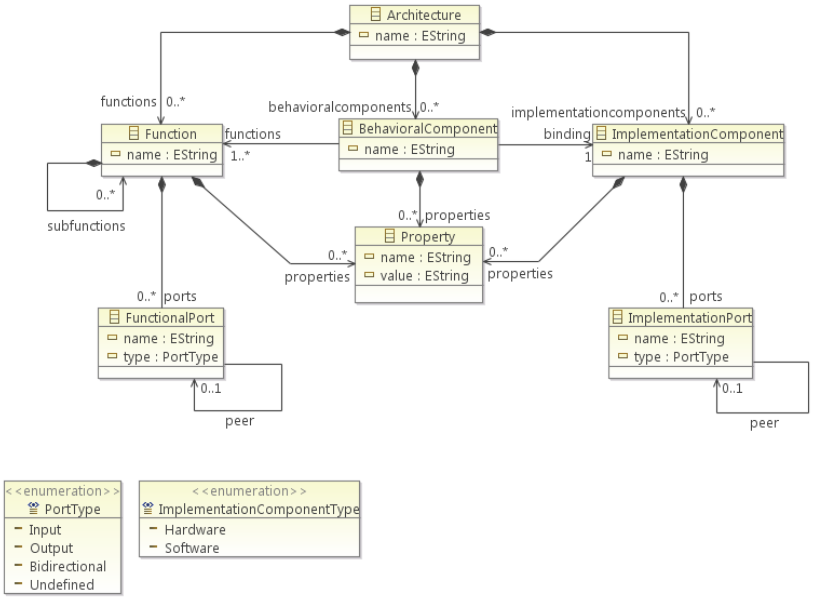


Figure 8 - Definition of the SDMS data model

3. The third step: Implement the data model

The implementation and the usage of the SDMS data model are still under experiment. The first results show that the SDMS data model facilitates the sharing of the common data between the different modelling languages used; in this case Altarica[12] and AADL[1].

b. The FCS (Flight Control System) Monitor change management scenario

The RTP principles and services have been also validated through a "change management" scenario that has been applied on the FCS Monitor application design from AleniaSIA.

This application implements Flight Control System Monitor functions for the Rudder sub-system of an executive Aircraft:

In this context the main objective was the definition and the implementation of a change in the application requirements. This change shall then be validated by proper test cases. The major steps of the expected workflow are represented in figure 9.

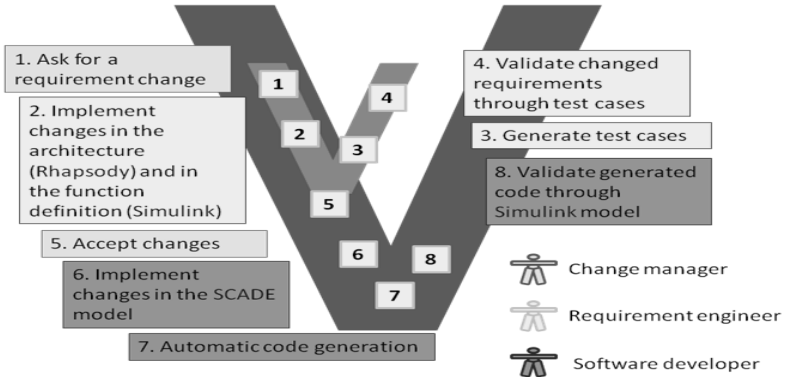


Figure 9: – Scenario workflow

The objectives we had in applying the RTP principles and services to our scenario were to:

- Reduce the human effort required in the left side of the "V-cycle" process
- Define requirements in a more formalized way in order to reduce errors in the system design
- Coordinate efficiently the different actors involved in the project by providing them both static and dynamic guidance about the tasks to be performed.

The identified tool chain to support this scenario is constituted by both design tools and RTP services covering process steps:

- Services: Link Repository; ModelBus [4]; RTP Desktop.
- Tools: IBM Doors[5]; IBM Rhapsody[6]; Mathworks Simulink[7]; Esterel SCADE[8].

Moreover, software requirements have been defined splitting the information in two models: a SysML[9] model which details the function execution order, and function input / output signal format, a Simulink model which details the functions. The software requirements written in formal language can be executed for validation purposes. The validation of the requirements of the FCS Monitor application needed a co-simulation of the logic defined by the monitoring algorithm and by the external environment. The algorithm and external system have been modelled in Simulink environment.

The SCADE design tool was used to replicate the Simulink requirement model. It was also used to generate code to be embedded in a Simulink block as an s-function. In order to demonstrate that the SCADE model and the Simulink block have the same behaviour, the SCADE model was uploaded in the Simulink model and supplied with the same monitor function input.

This method allows the System and Software engineers to validate their artefacts (requirement and code) in the same environment.

Coming back to the scenario workflow, we have to point out that the tasks performed in the framework of our evaluation have been specified through the SPEM [3] formalism, by selecting and customizing the proper Cesar Practice Library items. They represented the basis for originating the enactable BPMN[11] tasks and collaboration process.

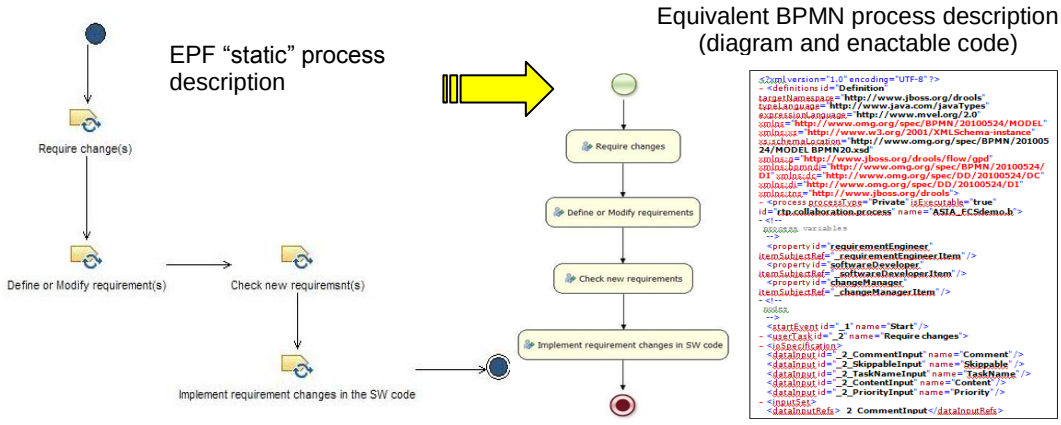


Figure 10: From SPEM[3] to BPMN[11] process description

The RTP Desktop application was used to get continuous guidance to the users about the tasks concretely driving the development activity.

Desktop views provide both the static description of the process and the guidance for each active task. In static description the roles are not associated to users, while in the active task view the roles and tasks are associated to actual users.

By integrating Model Bus Repository and Link Repository views, the RTP Desktop provides users with a unified access to artifacts and links created during the development tasks.

A lesson learned from this experiment is that a balanced definition of the process is a key-element for an effective use of the guidance functionality. In particular, for efficiency, the creation of short task chains that can be continuously re-used during the product development and for different projects is recommended.

The “change request process” moves in this direction, since the request for changes is continuously invoked during the product development. The tasks that made up this process fragment are basic, so that they can be found as basic blocks in a library.

Besides, we can mention that using the RTP approach and services it is possible to:

- Derive a development process from cross-domain background by tailoring it, thus enhancing reusability of practices and conformance to standards.
- Create an effective and unambiguous description of the engineering process through proper formalism. The RTP Desktop integrates this knowledge and provides an intuitive-user interface, continuously guiding the user.
- Represent an effective process knowledge base for customisation by each company thanks to the CESAR Practice Libraries.
- Support the validation of the requirements and accelerate the failure origin detection.
- Anticipate the validation of the requirements and the generated code by using simulation.
- Improve impact analysis and change management process by artefacts traceability.

Conclusion

In this paper, we described the concepts that allow customizing an engineering tools execution platform. The two main drivers of this parameterization are: the development life cycle chosen by the industrial company and the necessary data for specifying the different viewpoints of a product in a model-based approach.

A proof of concept of the relevance of these customization principles was demonstrated on two different use cases, the simplified doors management system concerning the system engineering data model construction and the flight control system for selecting the appropriate tools environment from a model of the system life cycle activities.

Currently, the level of automation of tools that support the experiments has not yet the level of maturity required for an industrial deployment. However, this automation is reasonably considered as feasible in the future by utilizing and evolving the work of the CESAR partners.

Moreover, the capabilities (e.g. installation procedures) to adapt the software artifacts (selected tools and described system engineering data) to the hardware platform are missing. Notwithstanding, an important step forward was made in the understanding of the enablers necessary to develop an automatic tools execution platform builder.

In order to make an automatic (or semi-automatic) tools execution platform builder operational for industrial deployment, further studies in the frame of cooperative environments are required. However, this paper shows a possible way to reach such an ambitious objective.

References

[1] AADL: Architecture Description Language
[2] FIACRE: an intermediate language for model verification in the TOPCASED environment
[3]SPEM: Software Process Engineering Meta model, an OMG standard
[4]ModelBus : model-driven tool integration framework
http://www.modelbus.org/modelbus
[5]Doors : a requirements management tool for systems
http://www 01.ibm.com/software/awdtools/doors
[6]Rhapsody: collaborative environment for system development
http://www-01.ibm.com/software/awdtools/rhapsody
[7]Mathworks Simulink: platform for system simulation and Model-Based Design
http://www.mathworks.fr/products/simulink/index.html
[8]SCADE: environment for the development of safety-critical embedded software
http://www.esterel-technologies.com/products/scade-suite
[9]SysML: Systems Modeling Language
http://www.omgsysml.org
[11]BPMN: Business Process Modeling Notation
http://www.omg.org/spec/BPMN
[12]Altarica: high-level language designed for the modelling of systems
http://altarica.labri.fr/forge/projects/3/wiki/AltaRicaLanguage