Designing a System Engineering Environment in a structured way

Anna Todino
Ivo Viglietti
Bruno Tranchero
Leonardo-Finmeccanica Aircraft Division
Torino, Italy

Rubén de Juan Marín
Instituto Tecnológico de Informática
Valencia, Spain

Abstract— **Defining and designing a System Engineering Environment in a complex industrial context is not a trivial task. Experiences, best practices, internal rules, company procedures and applicable standards, all play a role in the definition of a development process covering the different phases of development of a complex product. Engineers from various industrial domains need to develop complex systems in accordance with System Development Life Cycle Processes, defined within domains themselves, to assure project success in terms of time-to-market and product quality. These success criteria require reducing the development costs and applying an appropriate Design to Cost methodology to the different engineering projects, and the use of best practices is a mean to achieve this goal. Best practices consist of cross-domain and/or cross-project engineering methods and their application in the development of systems, where a specific engineering method addresses a preferred workflow by means of tools, involved artefacts and required interoperability between tools. Since the choice of a development tool can have some influence on the resulting system design, the characteristics of such tool must be clearly understood before the selection is made, and often the selection of tools is demanded to the ICT department, that sometimes is not completely aware of the system development engineers' needs. The idea is to reduce the development costs by reducing the time required to select the tool-set using ICT skill, process knowledge and applying cross-project engineering methods in order to specify and select the best solution. In this paper is presented a solution that is one of the outcomes of the CRYSTAL Research Project, for selecting a System Engineering Environment that satisfies the development process needs and supports engineering methods, which are formalized through SPEM 2.0.**
This solution is based on the Platform Builder Modeler, which allows to instantiate and validate a System Engineering Environment that includes tool-chain, IT infrastructure and data, dedicated to accomplish a selected development process.

Keywords—development process; SPEM 2.0; System Engineering Environment; Engineering Methods; Engineering tool functions, IOS capability, Tool taxonomy, Tool Integration; Tool-chain

I. INTRODUCTION

Within the industrial domains, it is a common practice to have a well-defined development process in order to manufacture a given product. A product development process defines a workflow of activities and related work products. A product development process is always supported by a proper System Engineering Environment (SEE) that is a framework which defines software, in terms of software tools used for development, hardware where the SEE is deployed, and data in terms of work products, models and other information relevant to the specific activities defined in the product development process. So the definition and configuration of a proper SEE is an essential part of a development process. One of the difficulties lies in the fact that depending on the available resources and on the skills of process engineers, it is not always straightforward to define a SEE.

The CRYSTAL Platform Builder (PB) is a solution based on the availability of an engineering methodology and the related engineering tools and also on an already defined development process, to support the definition and configuration of an instance of the SEE to be used by company engineers to manufacture the product. This approach starts from an existing product development process, which is the combination of a development process and of a product specific use case, and from Engineering Methods (EM) applicable within the engineering methodology and defined during the CRYSTAL project. This article is mainly composed by two sections:

- Concepts and meta-models: this is an overview and specification of concepts and meta-models dealing with PB methodology
- Platform Builder Modeler: this is an overview of the implemented solution based on the specified methodology.

II. Concepts and Meta-models

A. *State of the Art*

Nowadays even if the definition and formalization of the system development environment is presented as a set of consolidated formal steps, for example included in System development plan document, at industrial level no meta-models are available to describe a SEE. The topic is challenging and this is also demonstrated by the several research initiatives in the fields; for instance DevOps - toolchain group [7] is a tentative to collaborate about tool-chain taxonomy, in order to classify tools, following software development life cycle phases. Tool-chain taxonomy is also addressed by CRYSTAL PB approach where standardization and formalization of tools and tool-chain information and features is one of the major goals. Another example is the SafeCer project (http://www.safecer.eu), an ARTEMIS JU project, which provides CTF (Certification Tool Framework) that is a software platform that aims to integrate the different tools in a unique framework and allows building a tool-chain for certification purposes.

While SafeCER allows building a tool-chain for certification purposes, the CRYSTAL PB Modeler allows instantiating and validating a SEE, that includes tool-chain, IT infrastructure and data, dedicated to accomplish a selected development process and it is based on tool descriptors that formalize tool information within CRYSTAL PB.

B. *PB Approach Workflow*

The PB approach is based on process activities definition and relevant EM to be applied. Generally in a system life cycle process, activities to be performed for developing a system are defined together with work products to be produced and roles skilled to perform activities. This information is used by the PB approach to instantiate a SEE, to support the identified process in terms of engineering methods selected for an identified use case context, and to select proper tools to be used in the identified engineering methods. This approach requires the definition of a proper meta-model to facilitate the description of the process and for enabling the mapping between process activities, engineering methods and functions provided by tools. While a development process describes activities to be performed in an engineering process, engineering methods describe how to perform the different activities addressing engineering tool functions to be applied. Fig. 1 shows the complete workflow, identified in the CRYSTAL project, to configure and instantiate a SEE. Actually the PB scope encompasses only three phases of the complete workflow and implemented in the PB Modeler solution. For each phase different inputs and outputs are outlined:

- Tailoring phase: where the development process is described and formalized generating the Tailored Process.
- Configuring phase: where the tailored process is used to select tools and configure the best-fit tool-chain.
- Validation phase: where the tool-chain is evaluated against some business and technical constraints within the organization context.

Many different types of data must be classified and formalized in order to be used within different phases. This task was performed in the CRYSTAL project, where meta-models and data-models where defined. The result of the Tailoring phase is a Tailored Process, describing activities to be performed and engineering methods to be applied to develop a product according to the specified Use Case. The Tailored Process will contain information required to enable configuring a SEE using the PB Modeler. In the SEE configuring phase, using a catalogue of the available tools and their functionalities, a PB user will be able to define the best SEE solution for the Tailored Process in the given context. The Validation phase facilitates the configuration of a consistent and usable SEE. In the validation phase the SEE configuration shall be adapted to satisfy requirements and constraints for instantiating the SEE on the company IT infrastructure, thus producing the optimal solution in the provided company/organization context. Project data encompasses all information to set-up the SEE for a specific Tailored Process and project and it addresses mainly information relevant to user management and product management.

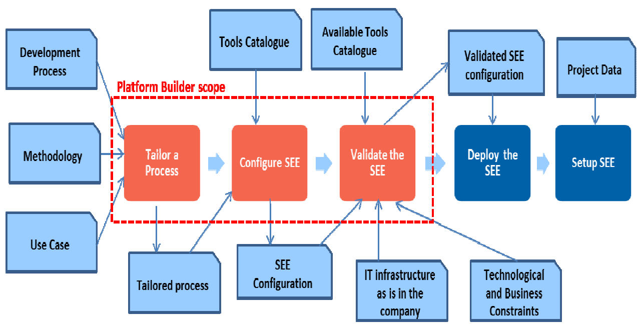

Fig. 1. SEE configuration and instantiation workflow

C. *PB Conceptual Meta-model*

Data and information, involved in the PB workflow, are represented in the PB conceptual meta-model (Fig. 2), where elements and their relationships for configuring the SEE are depicted. Some elements are aggregation of formalized data and they are defined as *catalogues*: Activity catalogue, Engineering Methods (EM) catalogue, Tool catalogue and Engineering Tool Functions (ETF) catalogue. All identified catalogues were populated to be used in different PB phases. Other PB elements specify data needed for PB method implementation. Herein each element of PB conceptual meta-model is defined. The main input for PB Modeler is the *use case* defining the project and the system to be developed in a defined business context. This includes also work products and activities to be performed in respect of business process and it is usually documented in natural language and not formalized. Use case information, relevant for the PB scope, is formalized in a *process*, which describes *activities*, *roles*, *artifacts* and *engineering methods* to be applied.

The process, herein also called *Tailored Process*, is used to describe all activities and engineering methods to be applied for the business context and for the specific use case. An *activity* is a concrete work identified within a development process and it represents a general unit of work assignable to specific *user,* with a *role*, able to perform it. An *activity* is a set of actions that consume time and resources and whose performance is necessary to achieve, or contribute to, the realization of one or more outcomes (*artefact*). An activity depends on the identified system life cycle process and it is specified within system development process. Indeed *Tailored Process* specifies which activities are selected for the specific use case, and the system development process, indicated by the

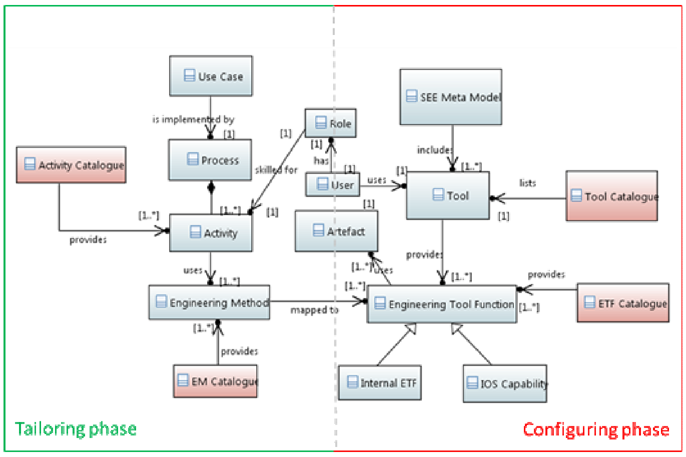

Fig. 2. PB conceptual meta-model

use case, is tailored with selected activities and detailed with Ems. While activity describes a task within a process, *EM* specifies how the activity has to be performed and in detail it describes a method which can accomplish a given identified result by defining steps to perform it and also artefacts that are produced or consumed. An Engineering method is formalized using ETFs that should be applied to satisfy it. On the other hand, a *tool*, that is any software application, can be described through the functionalities that it provides. A set of tool' functionalities, also said *ETFs*, are used to define a tool in a formalized way and this helps to select a tool to satisfy EMs.

The core of the PB method is the mapping between engineering methods and engineering tool functions. In order to fulfil this mapping, the Engineering tool function taxonomy is crucial. Indeed engineering method will be specified using engineering tool function as well, in this way the mapping between engineering methods and engineering tool functions make semi-automatic the selection of tools. Engineering tool function identification and categorization was made in extensive but not exhaustive way in the CRYSTAL project. For different examined use cases, engineering tool functions were listed and formalized in *ETF catalogue* to be recognized in the definition of *Tool Catalogue* and *EM catalogue.*

The SEE model [4] is configured using both tool information and tailored process information, and it is a collection of descriptors of tools and their relationships (tool-chain) through interoperability. In this article, interoperability stands for Interoperability Specification capability (*IOS Capability*) which is a specialization of ETF.

D. *Process Activity and EM classification*

The tailored process has to be defined in order to formalize the development process taking into account EMs to be applied during a system development. In the PB method this tailoring phase is facilitated by categorization of activities and EMs. Analyzing the needed information to be used for tailoring phase, the PB method has adopted the Software & Systems Process Engineering Meta-Model (SPEM) and identified some SPEM elements to formalize a process in the PB scope using a *reference method library* [1]. The tailored process, defined as a *delivery process*, encompasses *activities,* which groups EMs. Activities and EM are related to task element in the SPEM meta-model.

The CRYSTAL reference method library includes activities and engineering methods to be used during the tailoring process phase. The activities and EMs categorization was based on different domain use cases. The difficulty to make this classification and definition depends on the fact that the definition of activity is interpreted by engineers. Basically, using the activity definition and the SPEM task definition element, an activity is classified using a *name* and a *description* [5], considering that an *activity* describes a task to be performed within a system engineering life cycle process. Also for EM classification (refer to Table 1) is used the SPEM task element but in this case to describe it, the EM definition is applied: an EM describes tasks to be applied to satisfy an activity with support of tools.

TABLE I. EM

Engineering Method		**Example**
Name	Name of the EM as unique identifier. (Verb + Object)	Insert Requirements
Description	Short description about the EM to be applied.	Identified requirements are inserted and described into a Requirements database.
Work Products (Artefacts)	List of Artefacts handled by EM.	• Database-of- requirements • Textual-description
Guidance	Generic type of tool.	Requirements repository tool
Steps	List of steps to perform the EM.	• Create a new requirement • Describe requirement

Using this classification of activities and EMs a reference method library can be populated and it is considered as the activity and EM catalogue.

E. *ETF taxonomy*

As anticipated in the PB conceptual meta-model paragraph, the engineering tool function taxonomy is crucial for the PB Modeler. To help a PB Modeler end user to configure a SEE, the selection of tools is facilitated through the mapping between ETF provided by tools and those ETF required by engineering methods.

An ETF is a detailed function of a task that will be performed by means of a tool. Analysing use-cases proposed in CRYSTAL and engineering methods, a sort of ETF classification was identified. ETF has to be identified in a unique way and it can be used in different tools and also required by different engineering methods. Since a definition of ETF is necessary, we have identified different data to be specified to characterize ETF, and Table 2 shows the main attributes to be specified. In the PB conceptual meta-model, ETF are specialized as *IOS capability* and *internal tool function*. The difference between IOS capability and internal tool function is the type of ETF: if it is a proper internal function or if the function addresses interoperability with another tool. An ETF is classified as internal tool function if it manages data and it is not related directly to adapt data to provide/consume to/from another tool. It is any functionality provided by a tool that performs a functional operation within tool itself.

On the other hand, IOS capability is a ETF that implements interoperability with a different tool, in order to adapt data, to be provided or consumed, and that will be managed by another tool. Every type of interoperability between tools can be considered and interoperability specification can be defined for each kind of tool. Within the CRYSTAL project, IOS is referring Open Services for Lifecycle Collaboration (OSLC) specification [3].

TABLE II. ETF ATTRIBUTES

Name	Description
ID	Unique identifier for the ETF
Name	Name of the ETF
Description	Short description about the ETF.
Type	This field states if this ETF is an IOS service or an internal tool function.
Artefacts	Specify artefacts handled by ETF
Tool-class	Tools that can provide this ETF

F. *Tool and Tool-chain Taxonomy*

In the PB scope, tool properties formalization is important to facilitate tool selection. Indeed in configuring phase, tools that fit the EM requirements are presented and an end-user is able to select a suitable solution. Tool selection facility based on EM coverage, needs the tool categorization based on ETF classification. Different properties to describe a tool are listed in Table 3.

The formalization of a software tool based on ETFs shall be represented as a generic software (SW) component of tool-chain, refer to Fig. 3, where elements' relationships are depicted. Often, tool vendors give an overview of tools' functionalities and system requirements for its deployment in an informal way, through brochure or web site, and sometimes Tool vendors provide a formal description using manifest files. A tool descriptor collects in a formalized way all needed information to describe and to characterize software tool's functionalities, system requirements and extension relationships. This improves and facilitates the tool selection based on a non-ambiguous functionality.

TABLE III. TOOL PROPERTIES

Name	Description
Id	Unique identifier for the tool.
Name	Name of the Tool. Used for listing purposes.
Version	This field supports defining and distinguishing between different versions of the same Tool.
Vendor	Identifier of the Vendor.
Description	Short description about the feature provided by this Tool
Open-source	Indicates if the Tool is open-source or not.
License	Type of license
Extended-by	This field is used when the vendor of a tool knows that this tool is extended by another. Used for stating the use of IOS adapters (IOS service).
Extends	This field is used when the vendor of a tool knows that this tool extends the features of other tools. Used for stating the use of IOS adapters (IOS service).
SW-functions	List of ETFs provided by this tool (Internal tool function).
IT-requirement	Specifies the IT requirements of the tool.

A software component is defined to describe any kind of software, both system or application software: tool application, database, server, adapter, service, and so on. A generic software component represents information to identify any software implementation to be installed in a computer machine.

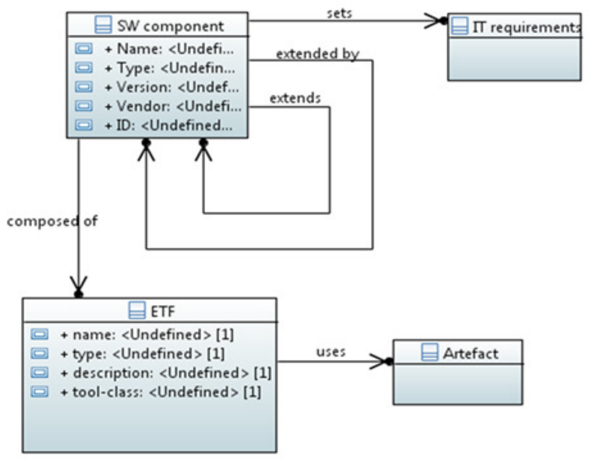

Fig. 3. Tool descriptor as a generic SW Component

G. *SEE Meta-model*

The PB approach mainly addresses the instantiation of a SEE which is configured through the PB Modeler tool based on, and extending Eclipse Process Framework Composer (EPF) [2]. All the needed information have to be described and formalized as specified by the PB approach in order to make the SEE configuration feasible in a semi-automatic and assisted way.

The SEE is a set of information that specifies: which process is used for system development, which engineering methods are applied within the selected process, which software tools and their characteristics are foreseen for the needed and identified tool-chain links to satisfy engineering methods. All these information are collected and specified in the SEE meta-model as shown in Fig. 4. A SEE model is the result of grouping and relating selected information from identified catalogues.

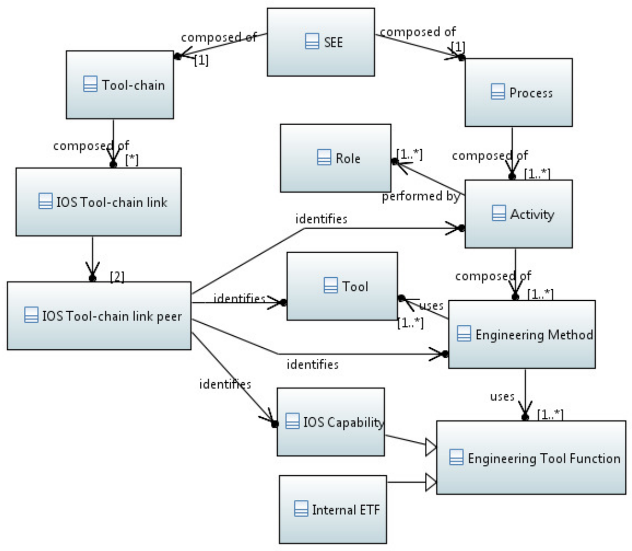

Fig. 4. SEE meta-model

In the SEE model, the tool-chain associated to the process is the main information. In the configuration phase, the tools selection is based on some required constraints that can be technical or business constraints. Technical constraints are relevant to tool characteristics, such as provided ETFs (IOS and internal tool functions), and IT constraints, such as availability of machines and network for SEE deployment and installation requirements (system requirements). Business constraints are relevant to the type of contract, if there exists, with Tool Vendor and company, the specific Tool type of contract: such as type of license, maximum number of users, and duration of contract. Technical and Business constraints are used in the PB Modeler to support tools selection in order to configure and validate the SEE model.

III. Platform Builder Modeler

The PB Modeler is the implemented solution of described PB methodology based on EPF Composer, for authoring the process using SPEM language, and some added plugins to configure and validate the SEE.

A. *Core Features*

Core features correspond to the main features of the PB Modeler tool implemented to address the PB workflow:

1. Tailor the process;
2. Configure the SEE;
3. Validate the SEE;
4. Generate the description of the SEE.

1) *Tailor the process*

This feature allows defining the resources that are needed for describing the development process, tailored for the specific use case. This is done using a graphical user interface to add data that encompass the PB meta-model for covering the relevant aspects. The PB meta-model specification is based on the SPEM meta-model, from which it selects relevant concepts for the Business Process, and it is enriched by concepts that are relevant to the SEE needs. This phase includes:

- The definition of activities sequence and activities themselves,
- The definition of work products,
- The definition of roles to perform activities and
- The application of engineering methods (EM) to perform an activity.

Generally these functionalities are already provided by a process authoring tool, nevertheless what is missing is a guideline and ontology to properly define activities and also engineering methods in a standard way. To facilitate the tailoring phase using an authoring tool, a CRYSTAL Reference Library was defined, containing at least the ontology about process activities (activity catalogue) and also for standardization of other meta-model identified elements, such as roles, work products and engineering methods (EM catalogue). The tailored process is the output of this phase and it is the main input of the following configuration phase.

2) Configure the SEE

The configuration phase groups all the functionalities that are necessary to formalize a SEE. This requires the enriched meta-model that includes the elements that are needed for defining a tool-chain, interoperability aspects and the deployment environment that is addressed in the IT infrastructure definition. The tool-chain is composed by software tools and the IT infrastructure is composed by network set-up, repositories and services that are the backbone for the selected tools. This phase includes the following functionalities:

- Import a tailored process (the tailored process as defined in the tailoring phase is used as input);
- Describe the general tool-chain in terms of tool function properties and IOS properties;
- Select tools, available tools in the business domain;
- Specify the IT infrastructure;
- Define repositories for the work products (data structure).

3) Validate the SEE

The validation of the SEE is performed on the configured environment that includes the tool-chain and the IT infrastructure and project data. In order to validate the configured tool-chain, the following inputs are required:

- the IT infrastructure as currently available in the company and defined using the PB meta-model;
- a selection of tools in the catalogue as available in the company (available tools).

The validation considers the tool-chain adaptability against the existing IT infrastructure, the evaluation of tools availability, and checking that defined IOS links are valid. The SEE has to be validated taking into account also the technological and business constraints (specific constraints on allowed vendors, standards, license type and if needed cost, in order to validate the data infrastructure, it is required to have the defined data structure in terms of work products and relevant repositories and to check its compatibility on the available IT infrastructure; e.g. the user management information includes the maximum number of users and which users are able to manage data, check the repository configuration and the network accessibility.

4) Generate the description of the SEE

After the validation, a SEE descriptor is generated. Such descriptor is a model and it contains information about the tool-chain and services to be deployed in the selected IT infrastructure. Project data useful to set-up the SEE will be generated starting from the required data infrastructure. Project data includes repository information and user management information.

B. Additional features of PB

The PB is mainly used for providing a model of a SEE, based on process activities to be performed in a product development. In addition to core functionalities, some features were as well implemented since they are propaedeutic to the Platform Builder goal. Indeed, such additional features produce outputs (EM, ETF and Tool catalogues, and also company IT infrastructure description) that are necessary as inputs to the different PB phases:

- Manage the Tool Catalogue: create a catalogue and describe tools in the catalogue applying the tool descriptor meta-model and tool properties.
- Manage the IT infrastructure: describe the IT infrastructure as it is currently in the company (available IT infrastructure) using the proper meta-model.
- Manage stakeholders: describe users and roles of the company's employees.
- Manage ETF and EM catalogues thought respective identified meta-models.

C. Operational View

In this section, tailoring, configuration and validation phases are detailed in order to describe user's operations. As depicted in Fig. 5, each identified phase in PB workflow depends on the previous phase: for example configuring phase depends on the tailoring phase which means that in order to configure the SEE, the Tailored Process has to be available before starting the configuration phase. Furthermore the validation phase depends on the result of configuration phase, because validation is done on the configured SEE. A Process Architect is in charge of tailoring the process activities and it has knowledge on its company business domain, while a SEE Architect is in charge of configuring the SEE and it has mainly knowledge on information technology department.

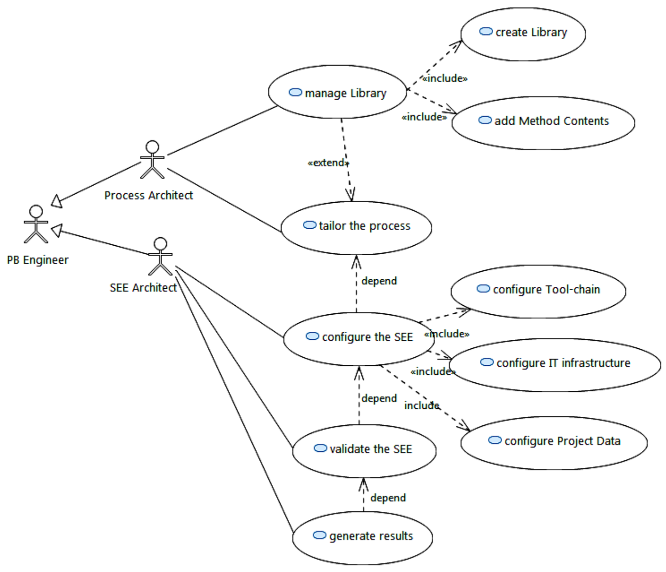

Fig. 5. UML use case diagram of PB

The *manage Library* use case is a pre-tailoring phase. It is part of authoring phase in order to prepare the CRYSTAL reference Library where are defined generic catalogues for activities and engineering methods.

1) Tailoring the process

This phase is intended to describe all the tasks needed to tailor the process in terms of operations to perform. Tasks to be performed are relevant to the description of a development process and to manage information in a formalized way to be used to configure the SEE. In Fig. 2 tailoring phase side, there are elements to define a development process using information of a process for developing a given product in a defined CRYSTAL use case. The Process Architect has to describe the development process and apply EMs during the tailoring process phase. To facilitate this task, when using a process authoring tool, process guidance, activities catalogue and EM catalogue are provided. During the tailoring phase, the following steps have to be performed:

- definition of the sequence of the process activities as well as the activities themselves;
- definition of roles to perform activities;
- definition of engineering methods applied to the process.

Any process authoring tool has to provide at least these steps that are also depicted in Fig. 6.

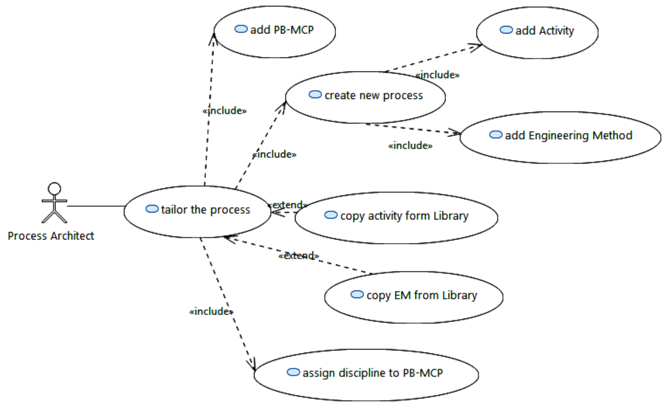

Fig. 6. UML use case diagram of tailoring a process

In order to facilitate the steps of the tailoring phase, it is necessary to provide a catalogue for each kind of identified elements in the process definition (such as process activity and roles) in a standardized way, and this is accomplished in the CRYSTAL reference library that specify an activity catalogue. Furthermore, an improvement concerns the definition of a catalogue of EMs to be applied. Generally, authoring tools provide interfaces to describe the product development process, but the Process Architect needs also guidelines to perform this task and information that can help to describe the process. Tailoring steps could be based on already existing authoring tools, extending them with the catalogue that define some needed elements addressed in the process. User interfaces, that should be defined to author a product development process covering also aspects relevant to SEE configuration, will be used to describe:

- Activities of the process and relevant elements
- Engineering methods to be applied.

2) Configuring the Platform

This phase encompasses all the tasks to be done by the SEE Architect for configuring the SEE in terms of operations to be performed. In Fig. 2 configuration phase side, there are all the elements and their relationships. The configuring phase is the core of PB method in terms of functionalities to describe the SEE. Basically, it is centred on the creation of a new SEE model related to a Tailored Process. After this step, the SEE Architect is able to configure the tool-chain (refer to Fig. 7) for each selected EM that is defined in the Tailored Process. This tool-chain configuration is based on mapping between EMs and ETFs and then between ETFs and tools which support identified ETFs. A SEE Architect has to select tools in order to satisfy the process activities and the applied EMs This task is supported by the *ETF Catalogue* and the *Tool Catalogue*. Furthermore, it has to describe the IT infrastructure (*configure IT infrastructure* use case) and formalize project data (*configure project data* use case) dealing with development process work products and roles/users of SEE. In the configuration phase the following activities have to be performed:

- Assign to the EM used in each process activity of the Tailored Process the required ETFs (IOS capability and internal ETF). This maps each EM to as many as required ETFs (*map EM to ETFs* use case)

- Select tools that will provide the ETF, thus becoming part of the tool-chain; the tool will be selected from the Tool Catalogue (*select Tool* use case)
- Establish the Tool interconnections (through IOS capabilities) in order to establish the needed tool interactions.
- Define the IT infrastructure needed for the tool-chain deployment, deriving it from the tools being used, foreseen users and needed services.
- Specify the permissions over the artefacts that each role has in each tool.

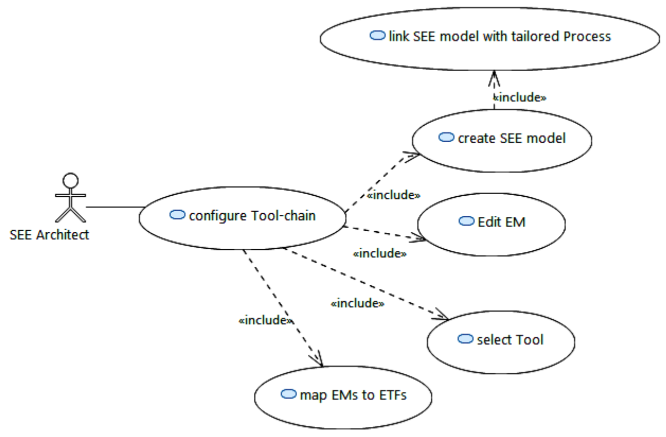

Fig. 7. Configuring the tool-chain

Once this process is finished, the SEE Architect has defined the tool-chain that will be used in the product development process and the IT infrastructure needs. IT infrastructure identification depends on selected tools to build-up a tool-chain, and some IT infrastructure properties could be described implicitly in Tool Descriptors. Fig. 8 shows what has to be identified for the IT infrastructure definition. IT infrastructure deals with all needed elements to build-up the backbone of the tool-chain and it encompasses services, repositories, network properties and user management.

The Data Structure identification, Fig. 5, depends on Artefacts required by the development process and it could be a simple list of Artefacts or more complex data structure. Data structure deals with work products involved in the process activities and project data includes data structure and also other information about roles/users and relevant properties to define access constraints, servers, repositories and also network information. For preparing project data, the SEE Architect needs information on IT infrastructure such as selected repositories and user access constraints.

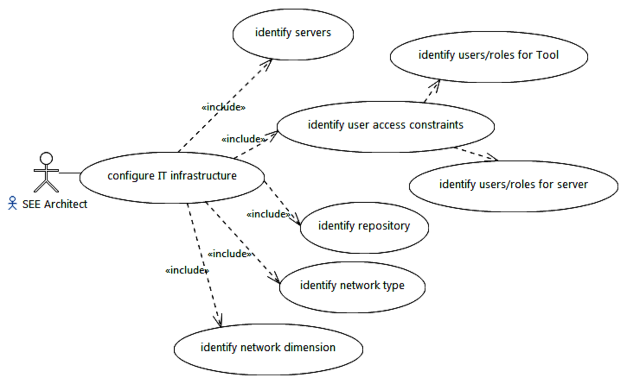
identify servers
identify users/roles for Tool
«include»
«include»
identify user access constraints
configure IT infrastructure
identify users/roles for server
SEE Architect
«include»
identify repository
«include»
«include»
identify network type
identify network dimension

Fig. 8. Configuring IT infrastructure

3) *Validate the SEE configuration*

In order to validate the SEE configuration that satisfies the development process objectives, the validation phase consists mainly in validating the tool-chain by comparing it with some criteria imposed by:

- the tailored process
- IT requirements and data infrastructure,
- Technological and business constraints such as applicable standards, specific constraints on allowed vendors, licence types and cost.

The SEE configuration validation could be implicitly made during the tool selection. Indeed some filter criteria can be applied to Tool Catalogue to propose a sub-set of tools that satisfy such criteria. For example, availability, license type and cost are filter criteria applicable to Reference Technological Platform (RTP) Tool Catalogue. Furthermore, IOS services coherence and IT infrastructure adaptability are validation criteria. The following figure shows validation tasks in order to validate the SEE configuration.

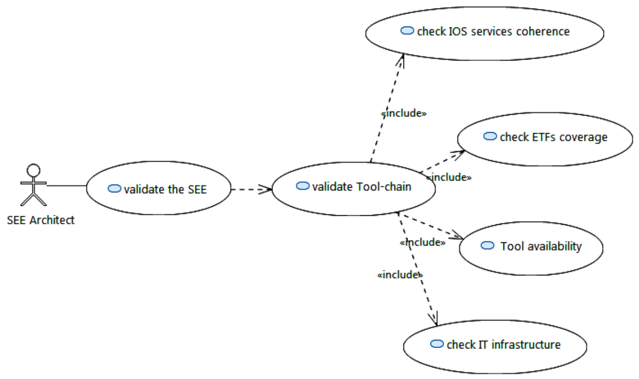

Fig. 9. UML Use Case diagram - Validation

The validation of a configured SEE implies:

- checking tool availability, which means selected tools are present and available in the company to deploy the tool-chain.

- checking the IOS services coherence: verifying that tool-chain links are valid.
- adapting, if necessary, the IT Infrastructure within the company organization: compare the required IT infrastructure's properties against the existing company's IT Infrastructure.

4) Generate the description of SEE

After these configuration and validation phases, the configured and validated SEE is the output that basically is a model to describe the configuration to be instantiated. The PB generates as output model the SEE configuration descriptor, which includes the tool-chain descriptor, the IT infrastructure descriptor and also a description of Project data.

IV. Conclusions

This paper describes the implemented Platform Builder method including identified and formalized meta-models for those relevant elements needed for SEE instantiation. It also describes the PB Modeler, that is a software tool that allows formalizing development processes and assisting SEE instantiation, which includes both process and tools information (such as Reference Technology Platform components). More in detail, the SEE model includes information related to:

- the required tools for the design of the desired product
- the Information Technology (IT) infrastructure where these tools are to be deployed
- data to be manipulated during the product development process.

The presented solution has been validated on many CRYSTAL industrial scenarios and it proven to be able to allow cost and complexity reduction for the activity of selecting the proper SEE for a given development process.

Acknowledgment

The research leading to these results has received funding from the research project CRYSTAL (Critical System Engineering Acceleration), funded from the ARTEMIS Joint Undertaking under grant agreement n. 332830 and from ARTEMIS member states Austria, Belgium, Czech Republic, France, Germany, Italy, Netherlands, Spain, Sweden, United Kingdom.
We would also like to thank our CRYSTAL partners among whom Pietro Braghieri (FBK), Luis Andreu (ITI), Guilherme Baungarten (OFFIS), whose active participation and collaboration in PB related subjects were essential for the achievement of the results.

References

[1] Object Management Group, "Software & Systems Process Engineering Meta-Model Specification Version 2.0," 1 April 2008. [Online]. Available: http://www.omg.org/spec/SPEM/2.0/PDF. [Accessed 30 March 2015].
[2] The Eclipse Foundation, "Eclipse Process Framework Project (EPF)," 2015. [Online]. Available: http://eclipse.org/epf/.
[3] OSLC, "Open Services for Lifecycle Collaboration," 2015. [Online]. Available: http://open-services.net/.
[4] "CRYSTAL deliverable - Meta-model specification - V2," 2015. [Online].
[5] "CRYSTAL deliverable – Platform Builder specification - V2," 2016. [Online].
[6] "CRYSTAL deliverable – Platform Builder prototype - V2," 2016. [Online].
[7] DevOps - toolchain group http://code.google.com/p/devops-toolchain/