Critical SYStem Engineering AcceLeration

Meta-Model specification V2
D602.012
<table>
<thead>
<tr>
<th>Project</th>
<th>CRYSTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grant Agreement No.</td>
<td>ARTEMIS-2012-1-332830</td>
</tr>
<tr>
<td>Deliverable Title</td>
<td>Meta Model Specification V2</td>
</tr>
<tr>
<td>Deliverable No.</td>
<td>D602.012</td>
</tr>
<tr>
<td>Dissemination Level</td>
<td>PU</td>
</tr>
<tr>
<td>Nature</td>
<td>R</td>
</tr>
<tr>
<td>Document Version</td>
<td>V2.0</td>
</tr>
<tr>
<td>Date</td>
<td>2015-03-05</td>
</tr>
<tr>
<td>Contact</td>
<td>Guilherme Baumgarten</td>
</tr>
<tr>
<td>Organization</td>
<td>OFFIS</td>
</tr>
<tr>
<td>Phone</td>
<td>+49 441 9722-176</td>
</tr>
<tr>
<td>E-Mail</td>
<td><a href="mailto:guilherme.baumgarten@offis.de">guilherme.baumgarten@offis.de</a></td>
</tr>
</tbody>
</table>
AUTHORS TABLE

<table>
<thead>
<tr>
<th>Name</th>
<th>Company</th>
<th>E-Mail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anna Todino</td>
<td>ALA</td>
<td><a href="mailto:anna.todino@alenia.it">anna.todino@alenia.it</a></td>
</tr>
<tr>
<td>Guilherme Baumgarten</td>
<td>OFFIS</td>
<td><a href="mailto:guilherme.baumgarten@offis.de">guilherme.baumgarten@offis.de</a></td>
</tr>
<tr>
<td>Rubén de Juan</td>
<td>ITI</td>
<td><a href="mailto:rjuan@iti.es">rjuan@iti.es</a></td>
</tr>
</tbody>
</table>

REVIEWERS TABLE

<table>
<thead>
<tr>
<th>Version</th>
<th>Name</th>
<th>Company</th>
<th>E-Mail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal Review</td>
<td>Stefano Russo</td>
<td>UNINA</td>
<td><a href="mailto:sterusso@unina.it">sterusso@unina.it</a></td>
</tr>
<tr>
<td>Internal Review</td>
<td>Pietro Braghieri</td>
<td>FBK</td>
<td><a href="mailto:braghieri@fbk.eu">braghieri@fbk.eu</a></td>
</tr>
<tr>
<td>External Review</td>
<td>Sytze Kalisvaart</td>
<td>TNO</td>
<td><a href="mailto:sytze.kalisvaart@tno.nl">sytze.kalisvaart@tno.nl</a></td>
</tr>
<tr>
<td>External Review</td>
<td>Grit Dudeck</td>
<td>IST</td>
<td><a href="mailto:dudeck@testingtech.com">dudeck@testingtech.com</a></td>
</tr>
</tbody>
</table>

CHANGE HISTORY

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Reason for Change</th>
<th>Pages Affected</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1.0</td>
<td>2014-01-31</td>
<td>Starting point of the second version of the Platform Builder Meta-Model: version 1.0 (composed of Meta-model Specification V1 and remaining open comments, remarks and doubts)</td>
<td>N/A</td>
</tr>
<tr>
<td>V1.0.1</td>
<td>2014-10-21</td>
<td>First restructuring/review of the document and update of the different diagrams/figures/tables</td>
<td>All</td>
</tr>
<tr>
<td>V1.0.1</td>
<td>2014-10-21</td>
<td>Complementary view of the Platform Builder process section moved from the CRYSTAL Platform Builder Meta-Model chapter to the Platform Builder Overview chapter</td>
<td>Chapters 2 and 4</td>
</tr>
<tr>
<td>V1.0.1</td>
<td>2014-10-21</td>
<td>Update of the Meta-Model diagrams and the description of their elements</td>
<td>Chapter 4</td>
</tr>
<tr>
<td>V1.0.2</td>
<td>2014-12-15</td>
<td>Update of different diagrams and text in all the sections of the document so that it reflects the current understanding about the Platform Builder Meta-model.</td>
<td>All</td>
</tr>
<tr>
<td>V1.0.3</td>
<td>2015-01-09,20</td>
<td>Chapters CRYSTAL Platform Builder Meta-Model and Platform Builder Catalogues were restructured:</td>
<td>Chapter 4 and 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Chapter 4 assembles now the main CRYSTAL Platform Builder Meta-Models and;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Chapter 5 gathers all the auxiliary descriptors, meta-models and catalogues.</td>
<td></td>
</tr>
<tr>
<td>V1.0.3</td>
<td>2015-01-09,20</td>
<td>Added new sections for specifying the high level and interoperability goals of the Platform Builder</td>
<td>Chapter 3</td>
</tr>
<tr>
<td>V1.0.3</td>
<td>2015-01-09,20</td>
<td>New terms were added to the Glossary and old ones were updated.</td>
<td>Chapter 6</td>
</tr>
<tr>
<td>Version</td>
<td>Date</td>
<td>Nature</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>V1.0.3</td>
<td>2015-01-09,20</td>
<td>The section about the compliance of CRYSTAL Platform Builder Meta-Model elements to SPEM became an Annex chapter.</td>
<td>Chapter 8</td>
</tr>
<tr>
<td>V1.0.3</td>
<td>2015-01-09,20</td>
<td>A section about the Platform Builder implementation goals was added to the chapter Platform Builder Overview.</td>
<td>Chapter 2</td>
</tr>
<tr>
<td>V1.0.3</td>
<td>2015-01-09,20</td>
<td>Section Relationship to SPEM – the CRYSTAL baseline meta-model added to the chapter Deliverable Scope.</td>
<td>Chapter 1</td>
</tr>
<tr>
<td>V1.0.4</td>
<td>2015-01-27</td>
<td>Adjustments and updates to the text and to the structure of the document were made.</td>
<td>All</td>
</tr>
<tr>
<td>V1.0.5</td>
<td>2015-02-02</td>
<td>Adjustments following Work Package review.</td>
<td>All</td>
</tr>
<tr>
<td>V1.0.6</td>
<td>2015-02-13,14,15,16</td>
<td>Adjustments following internal review.</td>
<td>All</td>
</tr>
<tr>
<td>V1.0.7</td>
<td>2015-02-26</td>
<td>Adjustments following external review.</td>
<td>All</td>
</tr>
</tbody>
</table>
## CONTENT

**ACRONYMS** ...................................................................................................................................................... 7

1 **DELIVERABLE SCOPE** .................................................................................................................................. 8
   1.1 DELIVERABLE ROLE .................................................................................................................................. 8
   1.2 RELATIONSHIP TO OTHER CRYSTAL DOCUMENTS .............................................................................. 9
   1.3 RELATIONSHIP TO SPEM – THE CRYSTAL BASELINE META-MODEL .................................................. 9
   1.4 DIAGRAMS NOTATION .............................................................................................................................. 10
   1.5 STRUCTURE OF THIS DOCUMENT .......................................................................................................... 10

2 **PLATFORM BUILDER OVERVIEW** ............................................................................................................... 11
   2.1 THE PLATFORM BUILDER IMPLEMENTATION GOALS ......................................................................... 11
   2.2 PLATFORM BUILDER BASELINE PROCESS ............................................................................................ 12
   2.3 COMPLEMENTARY VIEW OF THE PLATFORM BUILDER PROCESS ...................................................... 13

3 **CRYSTAL PLATFORM BUILDER META-MODEL REQUIREMENTS** ............................................................. 15
   3.1 HIGH LEVEL IMPLEMENTATION GOALS OF THE PLATFORM BUILDER AND RELATED DESIRED FEATURES 15
      3.1.1 Functionality ...................................................................................................................................... 15
      3.1.2 Interoperability Specification (within the Meta-Model) ...................................................................... 16
      3.1.3 SEE instantiation by means of an SEE configuration file ................................................................. 16
   3.2 IDENTIFIED PROCESS ELEMENTS ......................................................................................................... 16

4 **CRYSTAL PLATFORM BUILDER META-MODEL AND DESCRIPTORS** ............................................................ 21
   4.1 THE CRYSTAL PLATFORM BUILDER META-MODEL ........................................................................ 21
      4.1.1 The Core Meta-model ...................................................................................................................... 21
      4.1.2 The IT Meta-model .......................................................................................................................... 24
      4.1.3 The Data Meta-Model .................................................................................................................... 26
      4.1.4 The Software Component or RTP Component Meta-Model ............................................................ 27
   4.2 THE SYSTEM ENGINEERING ENVIRONMENT META-MODEL ................................................................... 30
      4.2.1 The System Engineering Environment Descriptor (Output of the Platform Builder Process) ....... 32
      4.2.2 The relationship between the different descriptors and the SEE ...................................................... 34

5 **PLATFORM BUILDER CATALOGUES AND DATA MODELS** ............................................................................. 35
   5.1 THE PLATFORM BUILDER METHOD CONTENTS PLUG-IN .................................................................. 35
   5.2 THE CRYSTAL REFERENCE LIBRARY ..................................................................................................... 36
      5.2.1 Activities Catalogue .......................................................................................................................... 37
      5.2.2 Engineering Methods Catalogue ...................................................................................................... 37
      5.2.3 Development Process (Guidance) .................................................................................................... 37
   5.3 CATALOGUES FOR TOOL DESCRIPTION ............................................................................................... 38
      5.3.1 Engineering Tool Function Catalogue ............................................................................................... 38
      5.3.2 Tools Catalogue .................................................................................................................................. 38

6 **GLOSSARY** .................................................................................................................................................... 40

7 **REFERENCES** ................................................................................................................................................. 43

8 **ANNEX 1 - COMPLIANCE OF CRYSTAL PB MM ELEMENTS TO SPEM** .................................................... 44
   8.1 MAPPING CONCERNING PROCESS ASPECTS (PROCESS STRUCTURE PACKAGE) .......................... 44
   8.2 MAPPING CONCERNING METHODS ASPECTS (METHOD CONTENT PACKAGE) ............................ 46
List of Figures

Figure 1.1: Link between identified items within the Platform Builder workflow ......................................................... 8
Figure 1.2: Structure of SPEM 2.0 ................................................................................................................................... 9
Figure 2.1: Platform Builder function ......................................................................................................................... 11
Figure 2.2: Platform Builder workflow....................................................................................................................... 11
Figure 2.3: System Engineering Environment configuration and instantiation workflow ........................................... 12
Figure 2.4: SEE configuration and instantiation activity diagram .............................................................................. 13
Figure 4.1: Identified required elements for the CRYSTAL Platform Builder Meta-Model ........................................... 22
Figure 4.2: The IT Infrastructure Meta-Model ............................................................................................................. 24
Figure 4.3: The IT Infrastructure Descriptor ............................................................................................................. 25
Figure 4.4: The Data Meta-Model ............................................................................................................................ 26
Figure 4.5: The Data Descriptor ................................................................................................................................ 27
Figure 4.6: The Software Component Meta-Model .................................................................................................... 27
Figure 4.7: The Software Component Descriptor ..................................................................................................... 28
Figure 4.8: The Tool-Chain Composition .................................................................................................................. 29
Figure 4.9: The Tool-Chain Descriptor ..................................................................................................................... 30
Figure 4.10: The SEE Meta-Model ............................................................................................................................ 31
Figure 4.11: Complementary SEE Meta-Model Showing Processes and Tool Artefacts .............................................. 32
Figure 4.12: The SEE Descriptor ................................................................................................................................ 33
Figure 4.13: Complementary SEE Descriptor Showing Process and Tool Artefacts ..................................................... 33
Figure 4.14: Relation between different descriptors and artefacts ............................................................................... 34
Figure 5.1: PB Method Content Template ................................................................................................................ 35
Figure 5.2: Example of the CRYSTAL Reference Library ........................................................................................ 36
Figure 5.3: Example of ETF descriptor contained in the ETF catalogue .................................................................. 38
Figure 5.4: Relation between the different tool catalogues ....................................................................................... 39
Figure 5.5: Example of Tool descriptor contained in the Tool catalogue ................................................................. 39
Figure 8.1: Taxonomy of classes defined in the Process Structure package (source: SPEM 2.0) ............................. 44
Figure 8.2: Mapping of CRYSTAL Platform Builder Meta-Model elements to the Process Structure package ........................................................................................................................................... 45
Figure 8.3: Taxonomy of Core Describable Elements (source: SPEM 2.0) ............................................................... 46
Figure 8.4: Mapping of CRYSTAL and SPEM MMs concerning the Method Content package ................................................. 46

List of Tables

Table 3.1: Platform Builder elements .......................................................................................................................... 18
Table 3.2: Platform Builder descriptors .................................................................................................................... 20
Table 5.1: The CRYSTAL Reference Library or the Platform Builder Catalogues ....................................................... 36
Table 5.2: The packages that group the Method Content elements within CRYSTAL Reference Library ........... 37
Table 5.3: The CRYSTAL Reference Library or the Platform Builder Catalogues ..................................................... 38
Table 6.1: Glossary Terms and Definitions ............................................................................................................... 42
## Acronyms

The first column acronyms stand for the text in the second column:

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMM</td>
<td>Common Meta-Model</td>
</tr>
<tr>
<td>EM</td>
<td>Engineering Method</td>
</tr>
<tr>
<td>EPF</td>
<td>Eclipse Process Framework (Project)</td>
</tr>
<tr>
<td>ETF</td>
<td>Engineering Tool Function</td>
</tr>
<tr>
<td>IOS</td>
<td>InterOperability Specification</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>MM</td>
<td>Meta-Model</td>
</tr>
<tr>
<td>PB</td>
<td>Platform Builder</td>
</tr>
<tr>
<td>RTP</td>
<td>Reference Technology Platform</td>
</tr>
<tr>
<td>SEE</td>
<td>System Engineering Environment</td>
</tr>
<tr>
<td>SPEM</td>
<td>Software &amp; Systems Process Engineering Meta-Model</td>
</tr>
<tr>
<td>UML</td>
<td>Unified Modelling Language</td>
</tr>
</tbody>
</table>
## 1 Deliverable Scope

Industrial experts from different domains agree on the necessity to enhance and simplify the specification and deployment of product design and development environments. CRYSTAL aims to define the so called System Engineering Environment (SEE) framework as an environment where a tool-chain is instantiated, taking into consideration the IT infrastructure and project data, and set up in order to perform a desired process needed in a specific project scope. In all cases, the SEE is a Collaborative Engineering Development Environment. From CRYSTAL project point of view, the SEE is an embedded system product development environment and the Platform Builder is a solution for realizing the instantiation of the SEE based on configuration information of a specific development process. The main objective is then to define innovative solutions and methods in order to improve the instantiation of this System Engineering Environment through the definition of a System Lifecycle Development and Management Process (Business Process or Development Process).

The present document defines the CRYSTAL Platform Builder Meta-Model concepts for a correct System Engineering Environment configuration and instantiation. Thus, the CRYSTAL Platform Builder Meta-Model must be able to detail the development process to be instantiated in a way rich enough as required by companies, including used tools, their interactions, needed project data and required IT infrastructure. Thereby, CRYSTAL Platform Builder Meta-Model addresses all the concepts that are necessary for covering the aspects exposed in the previous sentence as well as concepts concerning tool description, providing thus a standard for tool providers. Figure 1.1 depicts the main information types that are covered by the CRYSTAL Platform Builder Meta-Model. The orange element in this figure (the System Engineering Environment) is the central element in the Platform Builder approach and depends on further identified objects: Activities, IT Infrastructure, Project Data, Tools and Roles.

![Figure 1.1: Link between identified items within the Platform Builder workflow](image)

### 1.1 Deliverable Role

The objective of this document is to describe a meta-model which denotes relevant concepts for the SEE configuration and instantiation. The representation of these concepts in the meta-model level is necessary in order to have a common understanding of the System Engineering Environment from both industrial and tool vendor partners’ point of view, facilitating the SEE instantiation process. This document contains data requirements for SEE configuration and Platform Builder tasks and it represents the main input for the Platform Builder specification and prototyping tasks.
1.2 Relationship to other CRYSTAL Documents

This document can be seen as the starting point for the Platform Builder implementation work i.e. an input in terms of requirements that are relevant for SEE configuration and instantiation as well as an input for the Platform Builder specification document (D602.021) and for the prototyping task (D602.031). It is the second version of the Platform Builder Meta-Model, the first version being its starting point (D602.011).

1.3 Relationship to SPEM – the CRYSTAL baseline meta-model

The meta-model which was selected as baseline for the CRYSTAL Platform Builder Meta-Model was SPEM 2.0. Its specification can be found under the chapter References together with further information about it. SPEM stands for Software & Systems Process Engineering Meta-Model and, as its name makes explicit, it is a process meta-model with focus on engineering processes. SPEM has been selected because it is widely used for process definition, becoming a de facto standard that allows companies to define highly personalized processes. In this way, SPEM can represent the basic items that compose processes, which are used to represent some of the elements identified in the PB Meta-Model. Figure 1.2 shows SPEM structure and the relationship between the different elements/packages which compose the whole specification:

![Figure 1.2: Structure of SPEM 2.0](image)

SPEM general idea is that there are two main branches respectively called “Process” and “Method Content” which are based on a Core specification and are developed in a way that allows users to select the packages they need/want to use to define their processes, giving some freedom of application of the specification and avoiding useless implementation by reusing already accomplished work. There are also heterogeneous packages within SPEM structure which provide a way to integrate both branches concepts in order to multiply specification possibilities. Each existent package has its own elements which are consistent with the whole
specification as well as with each other i.e. they respect the specification hierarchy and are then naturally compliant to SPEM guidelines.

It is strongly recommended that the reader have good knowledge about the SPEM meta-model (see the chapter References).

1.4 Diagrams notation

The notation of the different diagrams presented in this document follows the CRYSTAL suggested notation for diagrams.

1.5 Structure of this document

In order to make clear the way this document should be studied, a short explanation of its structure is essential. It is composed by four main sections and one annex:

- **Platform Builder Overview** introduces to the reader an overview of the Platform Builder in order to make easier the understanding of the next sections and of the decisions that were made during their development;
- **CRYSTAL Platform Builder Meta-Model requirements** exposes the meta-model requirements, as inputs coming from the different partners, which were the basis for the creation of the meta-model;
- **CRYSTAL Platform Builder Meta-Model and Descriptors** presents the meta-model itself, explaining exhaustively its elements and their relationships, which aggregates the idea of a set of meta-models for the whole identified Platform Builder workflow, including tool and IT specification and validation of the resulting configuration.
- **Platform Builder Catalogues and Data Models** depicts the auxiliary Catalogues and Models that are necessary for the Platform Builder to work and be fully operational.
- **The Annex 1 - Compliance of CRYSTAL PB MM elements to SPEM** shows the mapping done between the CRYSTAL Platform Builder Meta-Model and the SPEM 2.0 specification.
2 Platform Builder Overview

The so called Platform Builder is a solution for improving the configuration and instantiation capabilities of the System Engineering Environment (SEE). Based on a Development Process description (see the Glossary), the Platform Builder aims to set up a SEE for a specific domain development project. The proposed SEE configuration has to be validated against the available tools and the IT infrastructure. Figure 2.1 shows in a simplified way the idea of the Platform Builder: what is the needed input in order to have the desired output.

![Platform Builder function](image)

Figure 2.1: Platform Builder function

In the scope of a development or business process (i.e. a business process definition based on the Meta-Model SPEM 2.0, which is an outcome of the project CESAR – see the chapter References for more details on SPEM 2.0 and the project CESAR, as well as the next chapters of this document) instantiated for a project/product giving a validated SEE configuration as result, the Platform Builder has to accomplish three main activities: tailor the process for a specific project and configure and validate the SEE. These ones are accomplished by means of several editors (parts of the Platform Builder) that define computer files based on the Platform Builder Meta-Models, as it will be seen in this specification. Figure 2.2 shows the sequence in which these activities are executed, the orange boxes being within the scope of the Platform Builder.

![Platform Builder workflow](image)

Figure 2.2: Platform Builder workflow

As it can be seen in Figure 2.1 and Figure 2.2, the scope of the Platform Builder is limited to define which are the required elements used to establish a proper and achievable SEE, starting with a development process (which is part of a business process) as basic input. Deployment and Setup of the SEE are out of the Platform Builder scope. The Platform Builder can also be called Platform Builder Modeller because a model is built at the end of the process workflow and all the functionalities provided by the Platform Builder address modelling activities.

2.1 The Platform Builder implementation goals

The Platform Builder is a software solution to build up a System Engineering Environment (SEE) which aims at supporting a Product Development Process. A SEE framework defines Software, in terms of tools used in the development, Hardware on which the SEE is deployed, and data in terms of work products and other information relevant to the specific activities defined in the Product Development Process.

The main inputs for this facilitator are the Development process definition and data for tailoring the process for a specific project (Use Case). Other important inputs depend on the company organization: they are the IT infrastructure and the list of tools available in the company. The Platform Builder produces as output the description of an SEE in terms of Tools and services, data to be set up and also a description of the tailored and validated IT infrastructure where the SEE and relevant data are deployed.

Therefore, the Platform Builder has to be able to provide the following features and functionalities in order to implement its desired functionality:

1. Tailor the generic process from methodology to the company process;
2. Configure the development environment i.e. the System Engineering Environment (SEE);
3. Validate the suggested SEE against the involved infrastructure information about Tool-Chain, IT infrastructure and Project Data;
4. Generate the description of the development environment (SEE configuration model);

There are also external features, which are not in the scope of the Platform Builder, but are rather used as input in the process. These external features are descriptions of:

1. A RTP Tools Catalogue;
2. An Available Tools Catalogue;
3. The current available IT infrastructure in the company/organization;
4. The Engineering Tool Functions (ETF) Catalogue;
5. The Engineering Methods Catalogue.

Some of these described elements will be later on used on the Platform Builder process. Implementation aspects are not the main subject of this document but instead all meta-models and descriptors related to these points are presented in the next sections.

### 2.2 Platform Builder Baseline Process

The preconceived instantiation workflow for the Platform Builder process is the first and main input for the creation of the CRYSYRAL Platform Builder Meta-Model. The process of configuring and instantiating a System Engineering Environment (SEE) in the scope of the CRYSYRAL Platform Builder can be seen in Figure 2.3:

![Figure 2.3: System Engineering Environment configuration and instantiation workflow](image)

From Figure 2.3, it is possible to notice that the basic idea is to execute a set of different activities one by one, having as input a development process and providing as output a validated SEE (considering only activities that are in the scope of the Platform Builder for the CRYSYRAL project). The activities that are explored in the scope of the Platform Builder, as exposed before, are to tailor the process and to configure and validate the SEE. The involved activities can be better depicted as follows:

1. Tailor the process for a project: this step consists in identifying and defining the Process Activities and their sequence, the Artefacts (Inputs, Outputs, etc.) used in each Activity, the Engineering Methods to be used and the Roles (define who uses the artefacts and his user rights access).

---

1 The light blue boxes in Figure 2.3 represent artefacts which are accessed or created by the different activities and therefore are defined in the chapter Glossary or in other sections of this document.
2. Configure the SEE: here, the local variety of Process Activities and their Engineering Methods are mapped to Engineering Tool Functions (provided in the Tools Catalogue\(^2\)), a model to represent work products in a repository is defined and IT Infrastructure constraints (as required by the tailored process) and the needed IOS properties are identified.

3. Validate the SEE: this last stage is responsible for mapping the Process Engineering Methods to Available Tool Engineering Functions (checking the needed Engineering Tool Functions against the Available Engineering Tool Functions), comparing the required IT infrastructure properties with the company current IT infrastructure and verifying if the required IOS properties are available.

Consequently the CRYSTAL Platform Builder Meta-Model has to cover all the required aspects in order to successfully perform sub-activities within each one of these main activities.

### 2.3 Complementary view of the Platform Builder process

In order to represent the SEE configuration and instantiation workflow process (Figure 2.3) as an activity diagram using UML notation which explicitly presents the main required elements for the process and their relationships, the activity diagram in Figure 2.4 was created. This new diagram can also be understood as a development of the diagram presented in Figure 2.3, for here some more high level elements are added and specific execution steps are set\(^3\).

---

\(^2\) Engineering Tool Functions, or ETFs, (provided by different tools) might not be available in the company. This configuration phase considers only what is necessary to perform the activities but does not consider if the Engineering Tool Functions are available. This evaluation is in charge of the validation phase. The Platform Builder cannot run without Engineering Tool Functions because, at first, the Engineering Methods are mapped to Engineering Tool Functions (from the ETF catalogue) and then these ETFs are mapped to the tools that provide these very same ETFs.

\(^3\) Indeed, Figure 2.3 and Figure 2.4 are not the same. Figure 2.3 shows the Platform Builder workflow in a static way, where objects and activities are identified for the whole SEE process. Figure 2.4 is dynamic, showing activities in a sequence, within the Platform Builder scope. They are different views of the same workflow.
In Figure 2.4, it is possible to recognize the workflow or sequence of activities that defines the SEE configuration and instantiation process of the Platform Builder. This diagram has a well-defined flow of activities related to the SEE setup, which starts, after the symbolic first point called “Project beginning”, with a definition of the required Business Domain Development Process (or simply Development Process) and has as output a configured and validated SEE. The Platform Builder Scope is presented in the centre of the diagram having as activities the tailoring of the process and the configuration and validation of the SEE. Some verification steps are also foreseen in order to check if all necessary tools or IT infrastructures aspects are available, but the activities of updating or completing these objects are not in the scope of the Platform Builder. The green boxes are representing the Platform Builder elements that are listed in the next section (Table 3.1). The ones that are in the boundary of the SEE configuration and instantiation process are meant to be out of the scope of the Platform Builder and are to be seen as external features/elements whose data can be manipulated without the use of the Platform Builder prototype (the implementation of the Platform Builder).

Comparing this activity diagram to the one presented in Figure 2.3, it is easy to see that some elements were added in Figure 2.4 in order to better describe the process (different activities) to be executed by the Platform Builder. These new elements (or aspects) are presented below:

- Evaluate Tool Catalogue is an activity placed between two of the main previous identified activities (Configure SEE and Validate SEE). In order to configure the SEE, it is required to know all needed Engineering Tool Functions for a given project (depending consequently on what is available in the RTP Tool Catalogue), while for its validation it is necessary to have them available in the company (it might be necessary to ask to tools providers to upgrade the functionalities in order to cover project needs);
- Tailored Process contains Activities and their related Process Engineering Functions;
- Here, IT Infrastructure changes its meaning depending on which activity it is referred:
  - IT Infrastructure seen as input of the activity Validate SEE is the existing or updated IT infrastructure available in the company;
  - IT infrastructure as required by the process is implicitly described in the object Configured SEE;
  - IT Infrastructure seen as input of the activity Deploy SEE is the validated IT Infrastructure;
- The “Re-evaluate” activities are only executed in case the different checks had negative outputs. If so, then they are to be understood as steps for complementing or adding the missing elements to their respective objects i.e. catalogues and IT Infrastructure.

This diagram was created using UML language which means that it is SPEM compliant i.e. the existing elements can be mapped to SPEM, as we will see in the CRYSTAL Platform Builder Meta-Model chapter.
3 CRYSTAL Platform Builder Meta-Model requirements

This section presents the identified meta-model high level goals and requirements, which were provided by the different domain and expertise partners. This information is relevant to the Platform Builder needs in terms of exchanging/managing/generating data.

3.1 High level implementation goals of the platform builder and related desired features

The CRYSTAL PB meta-model is the enriched meta-model that defines the Development process and needed SEE information. Sub-goals of the PB meta-model are related to functionality, interoperability and SEE instantiation, as follows.

3.1.1 Functionality

The operating environment for the Platform Builder consists of a framework where different applications that support User Interfaces coexist in order to edit and view data relevant to the SEE configuration. This framework is based on graphical user interfaces and on technology, like Java, that can be run in the majority of Operating Systems.

This framework also provides a friendly environment where the following features and related activities are to be conceived and executed (automatically or manually):

1. Tailor the process;
   a. definition of sequence of activities and activities themselves;
   b. definition of work products;
   c. definition of roles to perform activities and;
   d. application of Engineering Methods (EM) to perform an activity;

2. Configure the SEE;
   a. import a tailored process (the tailored process as output from the Tailoring phase is used as input here);
   b. describe the tool-chain in terms of Tools and Services (as part of the SEE configuration);
   c. select tools on the catalogue (available tools in the business domain);
   d. specify the IT infrastructure: derive the needed IT infrastructure (as part of the SEE configuration);
   e. define different kinds of repositories for the work products (data structure) e.g. various or distributed repositories;

3. Validate the SEE;
   a. check the tool chain adaptability against the existing IT infrastructure, the evaluation of tools availability and check the IOS properties and their compatibility;
   b. check the existence of required defined data structure in terms of work products and relevant repositories and check their compatibility against the available IT Infrastructure;

4. Generate the description of the SEE;
   a. generate a SEE descriptor containing information about the tool-chain and services to be deployed in the selected IT infrastructure;
   b. generate Project Data (repository and user management information) to set-up the SEE starting from the required Data Infrastructure.
3.1.2 Interoperability Specification (within the Meta-Model)

The CRYSTAL Platform Builder Meta-Model encompasses relevant elements for the development environment with regards to tool-chain definition and interoperability aspects.

As it will be seen in the next sections, the SEE Descriptor (or the SEE configuration model) is the main output of the Platform Builder and it defines a validated SEE configuration. The SEE descriptor contains information for describing the Tool-chain in terms of Tools and their properties, interoperability aspects and IT infrastructure constraints to be applied in order to have a correct and interoperable deployment of selected tool-chain.

3.1.3 SEE instantiation by means of an SEE configuration file

Taking into consideration both past classes of requirements and having in mind what is expected as output of the Platform Builder, it was agreed on the generation of a SEE configuration file that can be used for SEE instantiation purposes. Elements composing this SEE configuration file are also presented in the next sections. The SEE configuration file structure follows the related SEE meta-models and, since the deployment phase of the SEE is not in the scope of the Platform Builder, the assessment of it is not envisaged.

At the current phase of the implementation of the Platform Builder prototype, the SEE configuration file is a representation of the developed meta-model, with the required information for configuration purposes.

3.2 Identified process elements

Taking Figure 2.3 as reference, a necessary step to the refinement and definition of the requirements and sub-requirements that fulfil the Platform Builder Meta Model is to make explicit the breakdown elements and their relationships that compose the Platform Builder process. Table 3.1 details the different elements that are either used as input, output or are generated in this process, taking into consideration that in some cases elements comprise other elements. The definition of these relationships is important for the establishment phase of the meta-model.

<table>
<thead>
<tr>
<th>Element</th>
<th>Generated in</th>
<th>Used as input in</th>
<th>Description</th>
<th>Data Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development process</td>
<td>Development domain</td>
<td>Tailor a process</td>
<td>The development process provides generic information of a specific domain development process.</td>
<td>Development process descriptor</td>
</tr>
<tr>
<td>Project tailored process</td>
<td>Tailor a process</td>
<td>Configure the System Engineering Environment</td>
<td>The project tailored process details the process activities and their sequence, the artefacts being used, the participating roles and used engineering methods for a specific project. It comprises:</td>
<td>Tailored process descriptor</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Process Activities;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Engineering Methods;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Artefacts;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Roles.</td>
<td></td>
</tr>
</tbody>
</table>

4 The only point that is not envisaged in the project is the assessment of the deployment phase. The assessment of the tailoring, configuring and validating phases are envisaged.
<table>
<thead>
<tr>
<th>Element</th>
<th>Generated in</th>
<th>Used as input in</th>
<th>Description</th>
<th>Data Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project SEE Data</td>
<td>Tailor a process</td>
<td>Configure the System Engineering Environment</td>
<td>This element describes the artefacts and roles used in the tailored process. It is part of the Project tailored process.</td>
<td>Data descriptor</td>
</tr>
<tr>
<td>Engineering Tool Functions Catalogue</td>
<td>Part of the CRYSTAL RTP. Not defined within the Platform Builder. Continuously updated.</td>
<td>Configure the System Engineering Environment</td>
<td>This is the list of the engineering tool functions being used in the project. As a library, it is not the result of a single step, but rather it has to be continuously updated.</td>
<td>List of engineering tool function descriptors</td>
</tr>
<tr>
<td>RTP Tools Catalogue</td>
<td>Part of the CRYSTAL RTP. Not defined within the Platform Builder. Continuously updated.</td>
<td>Configure the System Engineering Environment</td>
<td>It contains information about the existing tools in the RTP. Each tool must define the provided engineering tool functions, the artefacts being used - in a per function basis - and IOS support. It comprises tools descriptors. It is continuously updated.</td>
<td>List of Tool descriptors (it is a local copy i.e. the one with updated or own internal tools which are the same as the Available Tool Catalogue)</td>
</tr>
<tr>
<td>Configured System Engineering Environment</td>
<td>Configure the System Engineering Environment</td>
<td>Validate the System Engineering Environment</td>
<td>SEE maps activities to engineering methods, specifies work products, identifies IT infrastructure constraints and the needed IOS properties. It comprises: - Tool Chain; - IT Project Infrastructure; - IOS properties.</td>
<td>System Engineering Environment descriptor</td>
</tr>
<tr>
<td>Tool Chain</td>
<td>Configure the System Engineering Environment</td>
<td>Validate the System Engineering Environment</td>
<td>It details the tools being used and their relationships, mapping outputs of engineering tool functions to inputs of other engineering tool functions. It also establishes the IOS needs. It is part of the SEE. It comprises: - Project Tools; - IOS requirements.</td>
<td>Tool Chain Descriptor</td>
</tr>
<tr>
<td>Element</td>
<td>Generated in</td>
<td>Used as input in</td>
<td>Description</td>
<td>Data Format</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---------------------------------------------------</td>
<td>-----------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>Tool Brick</td>
<td>Configure the System Engineering Environment</td>
<td>Validate the System Engineering Environment</td>
<td>It details tools functionalities and properties. It is part of the Tool Chain.</td>
<td>Tool Descriptor</td>
</tr>
<tr>
<td>IOS links</td>
<td>Configure the System Engineering Environment</td>
<td>Validate the System Engineering Environment</td>
<td>It details the IOS links needed in the process. These links are among the tools of the Tool Set.</td>
<td>IOS descriptor(it only details the IOS links which are needed in the process)</td>
</tr>
<tr>
<td>Available Tools</td>
<td>Configure the System Engineering Environment</td>
<td>Validate the System Engineering Environment</td>
<td>Information about the tools available in the company. For each tool it is required to define the provided tool functions, the artefacts being used - in a per function basis - and IOS support. It comprises a list of tools.</td>
<td>Tool catalogue</td>
</tr>
<tr>
<td>IT Infrastructure</td>
<td>Configure the System Engineering Environment</td>
<td>Validate the System Engineering Environment</td>
<td>This data provides information about IT infrastructure constraints for the project: needed repositories, security, safety constraints and network and machine requirements.</td>
<td>IT descriptor</td>
</tr>
<tr>
<td>IT Infrastructure currently available in the company</td>
<td>See previous row. Also, it is updated during the (re)evaluation of the IT Infrastructure</td>
<td>Validate the System Engineering Environment</td>
<td>This data must provide information about IT infrastructure constraints in the company: existing repositories, security and safety constraints and network information.</td>
<td>IT descriptor</td>
</tr>
<tr>
<td>Validated System Engineering Environment configuration</td>
<td>Validate the System Engineering Environment</td>
<td>Deploy the System Engineering Environment</td>
<td>This is the SEE obtained from the evaluation of the initial SEE against Available Tools and IT Company Infrastructure.</td>
<td>System Engineering Environment descriptor</td>
</tr>
</tbody>
</table>

Table 3.1: Platform Builder elements

Table 3.2 regroups the previous descriptors and extends the idea in order to derive additional descriptors to be used in the specification of the required elements, which compose the different phases of the Platform Builder workflow.

<table>
<thead>
<tr>
<th>Descriptor name</th>
<th>Description</th>
<th>Shall detail</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development process descriptor</td>
<td>It is a formalized description of Development Process.</td>
<td>It shall detail the process activities and their sequence.</td>
<td></td>
</tr>
</tbody>
</table>

Version Nature Date Page
V2.0 R 2015-03-05 18 of 47
<table>
<thead>
<tr>
<th>Descriptor name</th>
<th>Description</th>
<th>Shall detail</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tailored process descriptor</td>
<td>It is a formalized description of tailored process.</td>
<td>It shall be based on the CRYSTAL Platform Builder meta-model, enriching the Development Process Descriptor, linking activities to Project Data Elements and to Engineering Methods descriptors.</td>
<td>The linking information is part of the Tailored process descriptor.</td>
</tr>
<tr>
<td>Data descriptor</td>
<td>It is a formalized description of Project Data in terms of Artefacts and Roles.</td>
<td>This descriptor is part of CRYSTAL Platform Builder meta-model and contains two different elements: Artefact and Role. The set of information to be provided to each one of these elements is different.</td>
<td>Each main element of this descriptor must be linked to the activities in which it is needed.  When linked to the activity, it must be detailed as inputs, outputs or internal documentation.</td>
</tr>
<tr>
<td>Engineering Tool Function descriptor</td>
<td>This descriptor has to support the unambiguous identification of the engineering tool function. It also has to contain some description about it.</td>
<td>This descriptor shall detail the unique identifier, the name and the engineering domain of the engineering tool function.</td>
<td>Each engineering tool function must have a unique identifier at the RTP level. Those tool function descriptors are linked to the activities where they are used.</td>
</tr>
<tr>
<td>System Engineering Environment descriptor</td>
<td>This descriptor is the main output of the PB and consists of a description of the validated System Engineering Environment.</td>
<td>This descriptor details all the relevant elements that compose the validated System Engineering Environment.</td>
<td>This descriptor shall be defined to contain information, among other, about tool chain, data structure and IT infrastructure.</td>
</tr>
<tr>
<td>Tool descriptor</td>
<td>It contains:</td>
<td>Each tool descriptor must contain: Vendor, Version, IT requirements and Engineering domain. Moreover, it also contains: an Engineering Tool Function descriptor for each Tool function provided, and an IOS descriptor for defining Tool function IOS support. See Engineering Tool Function descriptor.</td>
<td>Both catalogues (RTP Tools and Available Tools) are a set of Tool descriptors. In the case of conflicting information between an Engineering Tool Function Descriptor and the information of the Tool descriptor where it is contained, the one defined at the Engineering Tool Function prevails. In this case IOS support must be provided at Tool Function Level.</td>
</tr>
<tr>
<td></td>
<td>• A list of Engineering tool function descriptors</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• IOS descriptor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Descriptor name</td>
<td>Description</td>
<td>Shall detail</td>
<td>Comments</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
<td>-------------</td>
<td>----------</td>
</tr>
</tbody>
</table>
| Tool chain descriptor | This descriptor is composed by:  
  - A list of Engineering Tool Function descriptors  
  - IOS descriptor | The Tool Chain descriptor in an extension of the previously obtained Tool Descriptors detailing information about the IOS needs concerning the Engineering Tools. | In this case IOS needs, if required, must be specified by activity – engineering tool function relationship level. |
| IOS descriptor | It represents aspects of interoperability based on the IOS specification between tools in a selected tool-chain | This descriptor serves either for defining IOS needs in a Tool Chain and defining IOS support in the Tools. The fields to be supported are: IOS specifications and versions. | When used in a Tool Chain, it is related to activity – engineering tool function pairs. When used in a Tool descriptor it is applied to the Engineering Tool Functions provided by the Tool. |
| IT descriptor | It defines a combined set of hardware, software, networks, and services/facilities where the tool-chain will be deployed. | All the important information concerning IT aspects and useful for characterizing the infrastructure where the tool-chain will be deployed. | |

Table 3.2: Platform Builder descriptors
4 CRYSTAL Platform Builder Meta-Model and Descriptors

In this section, the conceived CRYSTAL Platform Builder Meta-Model is presented. The idea is to get the requirements which were made available by the different partners and, from an existing baseline meta-model (SPEM 2.0 – see References and Annex 1 - Compliance of CRYSTAL PB MM elements to SPEM), develop the CRYSTAL Platform Builder Meta-Model. The chapter CRYSTAL Platform Builder Meta-Model requirements presented these requirements and goals whereas information about the baseline meta-model can be found under the chapters References and Annex 1 - Compliance of CRYSTAL PB MM elements to SPEM. The CRYSTAL Platform Builder Meta-Model is based on the SPEM Meta-Model and it is enriched in order to exhaustively support the specification and deployment of a fully integrated tool-chain.

The establishment process of the CRYSTAL Platform Builder meta-model followed the steps below:

1. Evaluation of the baseline meta-model in order to find out if it could be used, at least partially, as the start point for the conception of CRYSTAL Platform Builder Meta-Model;
2. Evaluation and classification of available requirements and different inputs coming from several partners;
3. Identification of relevant elements for the CRYSTAL Platform Builder Meta-Model and for the Platform Builder descriptors;
4. Initial mapping of previously identified elements to the baseline meta-model;
5. Gap analysis between specific Platform Builder needs and not-covered points from the mapping process (previous step);
6. Enrichment of the CRYSTAL Platform Builder Meta-Model in order to cover missing aspects from SPEM;
7. Consolidation of the meta-model.

Also, from the identified elements presented in Table 3.1 and the Platform Builder workflow itself (Figure 2.3), a deeper representation of the meta-model consisting of descriptors was developed.

The result of the Meta-Model and Descriptors creation process is presented in this chapter under the several following sub-sections.

4.1 The CRYSTAL Platform Builder Meta-Model

Based on collected requirements and on the Platform Builder workflow process, as well as taking into account the implementation aspects which come from the first version of the Platform Builder prototype (D602.031) and its specification (D602.021), the meta-models were further developed since the last version of the meta-model specification document (D602.011) and are now structured as a Core meta-model plus specific meta-models and descriptors for the diverse elements that require a specification on these terms. The diagrams in the next sections represent the identified required elements for the CRYSTAL Platform Builder Meta-Model and how they are related to each other.

4.1.1 The Core Meta-model

The Core Meta-model contains the high level elements used in the Tailoring and Configuration phases foreseen in the Platform Builder workflow (Figure 2.2). The Core Meta-model can be seen in Figure 4.1:
In Figure 4.1, it is presented the final version of the PB Meta-Model which was conceived after an iterative process, taking into consideration diverse feedback from all involved partners. This Meta-model is composed of 2 main phases (green and red squares in the diagram), which are respectively:

1. The tailoring phase: it corresponds to the formalization of the process in form of a SPEM compliant Development Process (upper side of the green square) AND to the mapping of the Process Activities to Engineering Methods, which are provided by the Engineering Methods Catalogue (lower side of the green square);

2. The configuring phase: it corresponds to the adaptation of the mapping of Engineering Methods to the Engineering Tool Functions, which are provided by the Engineering Tool Function Catalogue (lower side of the red square) AND to the assignment of Engineering Tool Functions and Tool Bricks, which are listed in the Tool Catalogue (upper side of the red square).

Following the study of the diagram presented in Figure 4.1, it is possible to see that several different elements are used to define the meta-model, having in mind that the Core Meta-model relates Processes (or Development Processes) with Tools (or Tool Bricks) by the relation between Process Activities, Engineering Methods and Engineering Tool Functions. These elements are defined as follows:

→ **Artefact**: it is any work product generated, modified or deleted by an Internal ETF or an IOS Service. Artefacts are described by means of internal tool (meta-) models.

→ **Engineering Method**: 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.

Engineering Tool Function: it is an Engineering Function representative of a task that a given Tool Brick is able to perform i.e. it is a detailed functionality that a Tool Brick is able to provide and that is specialized by Internal ETFs and IOS Services.

Engineering Tool Function Catalogue (ETF Catalogue): ETF Catalogue is a container of Engineering Tool Functions (descriptors).

Internal Engineering Tool Function (Internal ETF): it is any functionality provided by a Tool performing operations within the Tool itself i.e. it is an Engineering Function that is not related to the adaptation of data in form of provider/consumer patterns.

IOS Service: IOS Engineering Function or IOS Service is any functionality provided by a Tool Brick that is supported by the IOS e.g. an engineering function of a tool adaptor that manages data (i.e. an IOS Engineering Function) in order to:
- read and adapt data for use of the tool itself (consume data/artefact - the adaptor would be a consumer) or to;
- make data available to other tools (provide data/artefact - the adaptor would be a provider).

Process: a process (or a development process) defines process activities to be performed in order to develop a given system. In the scope of a given process, different activities can be identified and each one might be characterized by different sequences of Engineering Methods for it depends specifically on the selected use case.

Process Activity: as described in the ISO/IEC 15288, 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. A process activity understood from the point of view of the Platform Builder is any activity that composes or is part of a workflow, that is described in the Process i.e. a development process, that an User having a given Role is able to perform it and that uses Engineering Methods as an interface to the mapping to Engineering Tool Functions. It is also defined as actions to be performed in order to accomplish the process goal i.e. the development of a specific system/product.

Role: It defines a skill or function assumed by an actor performing an activity in the process.

System Engineering Environment (SEE): the SEE is a development environment where a tool-chain is instantiated in order to be used by Users to develop a given product. Considering the tailored process as input, a SEE descriptor has to be derived in order to contain the necessary information (tools information for deployment and interoperability as well as information about IOS services and IT Infrastructure that describes the environment where the Reference Technology Platform (RTP) Instances will be deployed) to set-up the SEE itself. The SEE descriptor has then to be validated by checking the tool-chain descriptors against the Company Available Tools Catalogue and by checking the IT Infrastructure descriptor against the organization available IT infrastructure.

Tool Brick: it is any Software Component such as Engineering and Management Tools, Services and Repositories that can be described in terms of their functionalities and that can be part of a System Engineering Environment by providing useful Engineering Tool Function(s) i.e. by supporting Engineering Methods to be used in a system development process.

Tool Catalogue: Tool Catalogue is a container of Tool Bricks (descriptors), as it can be seen in more detail in section Tools Catalogue. In the Platform Builder workflow which is exposed in Figure 2.3, there are two different categories of tool catalogues:
- Tools Catalogue and;
- Company Available Tools Catalogue. eventually

5 The information about the OSLC artefact that is eventually modified is available in the ETF Catalogue.
→ **Use Case**: it is a domain specific framework reflecting stakeholders objectives related to their businesses. It originates the requirements and process information and incorporates all the expectations concerning the Platform Builder outcomes (in the context of this document) i.e. it describes information relevant to a system to be developed in the company context and it is used to originate requirements and process information to describe and tailor the process itself. It is usually documented and available in form of textual descriptions.

→ **User**: it is the person who plays a Role and access and use the deployed System Engineering Environment. Its definition requires general information of the person and implies the choice of user management information in order to control access rights.

### 4.1.2 The IT Meta-model

The IT infrastructure can be seen as two different-in-time elements (IT infrastructure currently available in the company and IT infrastructure required for the project) and it deals with storing, retrieving, exchanging and manipulating data. IT infrastructure addresses also aspects that are relevant to data structure, data security and user access rights, these elements being identified in the configuration phase. IT Infrastructure properties have to be derived from the tailored process (e.g. services) and finalized in the configuration phase after tool selection in order to properly describe needed repositories and their locations, data views and security and safety constraints (access rights and access control level). IT infrastructure also depends on work products (mainly for data views and repositories) and roles (mainly for access rights and access control level).

The IT infrastructure (as required for the project) defines the IT infrastructure to where the tool-chain will be deployed. This IT infrastructure is the platform which provides the basis for the data structure, roles, applications and functionality being delivered to the end users in order to allow them to perform the required activities.

The **IT Infrastructure Meta-Model** can be seen in Figure 4.2:

![Figure 4.2: The IT Infrastructure Meta-Model](image-url)

---

**Version** | **Nature** | **Date** | **Page**
---|---|---|---
V2.0 | R | 2015-03-05 | 24 of 47
In this meta-model, IT deployment requirements are represented. It is possible to see that there are common elements between the IT infrastructure meta-model and the core meta-model, which are: the RTP Component\(^6\) (which is a Service\(^7\) or a Tool Brick) and the User. These elements are the link between both meta-models.

The IT infrastructure is defined to support the identified Tool-Chain which is composed by RTP Components. Each RTP Component, as a Software (SW) component, has IT requirements such as Machine Hardware (HW) and Software requirements. Therefore the IT meta-model describes relationships between Tool-Chain (composed by RTP Components) and IT infrastructure elements that supports the Tool-Chain itself. It identifies all relevant elements used to describe the IT infrastructure in terms of physical setup: network of machines that will be used to support the tool-chain against RTP Component/SW Component requirements and SEE Engineer choices during configuration phase.

### 4.1.2.1 The IT Infrastructure Descriptor

The IT Infrastructure descriptor contains information about the necessary IT infrastructure for deploying a given Tool-chain. This IT information includes necessary network and services in terms of security and safety constraints, data views and repositories. The IT infrastructure descriptor is used for setting up the IT infrastructure that is used for deploying the Tool-Chain. Its structure is shown in Figure 4.3:

![Diagram of the IT Infrastructure Descriptor](image_url)

**Figure 4.3: The IT Infrastructure Descriptor**

An IT infrastructure required to deploy a given tool-chain shall be derived from the tool-chain itself. Some characteristics required by the tools to be deployed are described in the SW Component Descriptor, which is described in the next sections. The requested data is the following:

- Physical network layout;
- Number of machines to be deployed;
- Needed services;
- Needed tools;
- For each machine: hardware characteristics, Operating System and software to be installed (related to necessary tools and/or services for the process).

This information comes from the IT infrastructure meta-model diagram (Figure 4.2). Moreover, the IT infrastructure descriptor outlines:

\(^6\) A RTP Component is the generalization of a Service or a Brick. In UML diagrams, the generalization (and the specialization) is a good representation of relationship for identifying a generic element. It is used in order to simplify diagrams and to have the relationships “hosts” and “uses” for the same element in a diagram.

\(^7\) In the Platform Builder scope, service means a tool brick or software used as a service.
• Roles (define who uses the artefacts and his user rights access);

• Tools constraints (IT infrastructure has to take into account tool constraints for tool constraints could also depend on Artefacts (which they use or access). Generally, tool constraints could concern hardware properties/system properties but also network properties to access data (artefacts). Depending on the artefacts that a tool uses, the IT infrastructure could have its network and repository properties changed, which would depend on the relationship between artefacts and tools relationship).

Information about the properties of artefacts (which and how the artefacts are used concerning data security levels, format for data views, repository location) are required for the definition of the IT infrastructure as well.

### 4.1.3 The Data Meta-Model

For handling the necessary data within the Platform Builder process i.e. its different phases, the Data meta-model presented in Figure 4.4 was developed⁸:

![Data Meta-Model Diagram](image)

The yellow boxes present complementary information about the meta-model elements. Using these data, it is possible to derive the necessary information for the initial IT requirements and Project Data.

#### 4.1.3.1 The Data Descriptor

The Data Descriptor specifies the Artefact element with their properties as well as the Role element defining its attributes as first identified in the Data Meta-Model.

During the tailoring phase, some data is defined as well as their properties e.g. maximum number of Users that are allowed to play a Role or permission levels for each Artefact. Furthermore, an Artefact can define the type of artefact or tool that supports it. This data is classified following the notation of the Data Descriptor, which is shown in Figure 4.5:

---

⁸ The Data Meta-Model is on the point of view of Artefacts, which is the central element of the diagram presented in Figure 4.4 and therefore relationships are identified around the Artefact element. In a Tailored Process, artefacts are defined, composing it. Tools use/manage Artefacts and Roles access Artefacts. It does not mean that the process interact directly with the artefacts.
4.1.4 The Software Component or RTP Component Meta-Model

A Software Component or an RTP Component is any service/brick tool that is able to compose a Tool-Chain. The Figure 4.6 presents the Software Component Meta-Model.
In Figure 4.6, IT Requirements refer to machine requirements, network requirements, safety/security requirements and required services.

Some relationships need a deeper explanation, as follows:

- **Extended by** is related to a list of brick identifiers;
- **Extends** is related to a list of bricks this brick extends;
- **Requires** is related to the IT Requirement which requires a Service (Services are described using the Software Component Descriptor itself);
- **Sets** refers to the fact that a Software Component sets IT requirements in order to be deployed (IT Requirements express it).

Also, a Software (SW) Function is any software functionality that a Tool/Brick or Service is able to provide.

### 4.1.4.1 The Software Component Descriptor

The Software Component Descriptor is used within the PB to allocate Engineering Methods and Software functions as well as to describe software implementation. Figure 4.7 shows its elements and their attributes:

![Diagram of Software Component Descriptor](image)

For the Software Component Descriptor, the following definitions are valid:

- **Software (SW) Function** is any software functionality that a Tool/Brick or Service provides;
- The Software Component Descriptor is used to describe any RTP Component (Service or Tool/Brick) which means any tool-chain component;
- **IT Requirements** define the dependencies and IT constraints of a RTP Component.
- **Resident SW** is a software that has to be installed within the machine that hosts the SW Component;
- **Services/remote SW** is a software tool that has to be installed on a server machine in order to be accessed by the machine that hosts the SW Component;
• *Network-deployment* describes a limited number of users concerning single access or concurrent access privileges;
• *Type* is a detailed Engineering Domain or a Service.

There are common aspects between Software Component Descriptor depicted in Figure 4.7 and the IT Infrastructure Descriptor from Figure 4.3 i.e. a relationship exists between the Software Component Descriptor and IT Requirements which are necessary for Software Component deployment. Considering the needed information used to define the IT infrastructure, the following applies for both diagrams:

• Repository and Services are Software Components and are described by Software Component Descriptors. The relationship "sets" covers these IT infrastructure elements;
• Network and Safety/Security aspects are covered by IT requirements meta-models elements as User and Network ones.

### 4.1.4.2 The Tool-Chain Descriptor

After defining a RTP component, it is possible to define a Tool-chain as a set of tools i.e. RTP components which are employed to a specific use and their interoperability aspects in the respective system lifecycle. The tool-chain has to be defined by the tool-chain descriptor that contains information about different tools and interoperability aspects between them. Tool information specifies the necessary data used to classify tools in terms of provided and consumed services which are full described by the engineering tool functions. Figure 4.8 represents the RTP component as the central element of a well-defined composition, which makes a link between the unitary tool features and the tool chain: the tool chain composition.

![Tool Chain Diagram](image)

**Figure 4.8: The Tool-Chain Composition**

Tool-Chain is configured selecting RTP components\(^9\) from a Tool Catalogue (see the chapter Tools Catalogue). Selected RTP Components provide Engineering Tool Functions that satisfy Engineering Methods.

---

9 The CRYSTAL RTP or Reference Technology Platform is a complete set of software tools from tool vendors that participate in the project CRYSTAL and that are to be integrated by means of the Platform Builder.
10 A RTP Component is the generalization of a Service or a Brick. In UML diagrams, the generalization (and the specialization) is a good representation of relationship for identifying a generic element. It is used in order to simplify diagrams and to have the relationships "hosts" and "uses" for the same element in a diagram.
The configured Tool-Chain Descriptor is then a list of RTP Components and their relationships (the tool links) that represents the Tool-Chain as required by the Tailored Process. Configured Tool-Chain Descriptor is the input for the validation phase. In other words, the Tool-Chain Descriptor is a collection of Software Component Descriptors and the respective IOS Tool Chain Link11, as presented in Figure 4.9, in the implementation point of view:

![Figure 4.9: The Tool-Chain Descriptor](image)

As way of example, a given IOS Tool Chain Link or its two IOS Tool Chain Link Peer could be represented as follows:

```
Activity_ID.EM_ID.Tool_ID.IOS_service_A_consumer<->Activity_ID.EM_ID.Tool_ID.IOS_service_A_provider
```

The Tool Chain and its relationship with the Tool Chain Links are further presented in the next section.

### 4.2 The System Engineering Environment Meta-Model

This section presents the System Engineering Environment (SEE) Meta-Model the way it was conceived vis-à-vis implementation aspects. It consists of diagrams which present the different elements used in the implementation of the SEE configuration and validation processes. For general information, the SEE configuration schema is a combination of XML files and folders structure. The whole SEE configuration schema is presented in this section as two different diagrams: the SEE configuration schema without processes and tools artefacts and a complementary schema presenting the different artefacts and their relationship with other SEE configuration elements.

The SEE meta-model is presented in Figure 4.10:

---

11 Following the OSLC principle where a tool publishes its data to n other tools, Figure 4.9 tries to represent which are the tool connections that are useful for a company in each Engineering Method of a process for a specified use case. Among, all the possibilities of interoperability links, it is only highlighted the ones that are interesting for the process i.e. the represented links are used to specify in which connections, among all the possible ones, the company is interested in for a given process.
In Figure 4.10, it is possible to understand the way the SEE is configured and of which elements it is composed. Also, it is easy to notice that the diagram is composed of two branches: the Tool Chain and the Process ones. It should be highlighted here that for interpretation purposes of this diagram, each depicted element is the representation of the real element in the SEE configuration file. Some relevant aspects/further information concerning the SEE configuration schema are the following:

- The Process branch is a representation of the EPF process (see the References for more information on EPF Composer);
- Tool Bricks provide Engineering Tool Functions (ETF) (not shown explicit here);
- An Engineering Method uses the Tools/Bricks which provide the its needed ETF;
- IOS Tool Chain Link Peer is only used for linking IOS Services;
- An Activity in the SEE configuration files points to the Engineering Method(s) (EM) it contain(s);
- An Engineering Method (EM) stores the information related to:
  - The Engineering Tool Functions (ETF) it uses;
  - Tools/Bricks that provide the ETFs used in this same EM;
  - The Steps (descriptive information about EMs).
The IOS Tool Chain Link contains 2 Peers (one consumer and one provider):

- Each Peer specifies the sequence: “Activity -> EM -> Tool -> IOS service”\(^\text{12}\):
  - The “Activity -> EM” service details in which part of the process this link is being used;
  - The “Tool -> IOS” service details the tool and service being connected.

It is also important to notice that in the Tool Descriptor of a Tool, the Tool vendor specifies the ETFs supported by a Tool / Brick\(^\text{13}\).

Once the general overview of the SEE configuration was given, it remains the relationship of the different artefacts to be explored. In Figure 4.11, this relationship is presented:

![Diagram of artefacts relationships](image)

Figure 4.11: Complementary SEE Meta-Model Showing Processes and Tool Artefacts

From Figure 4.11, it is possible to see two different kinds of artefacts: the Process Artefact and the Tool Artefact. A Process Artefact is a generic form of artefact and it is not the same as the Tool Artefact i.e. they have their own defined formats. Process artefacts can be identified when defining Process Activities and their implementation/realization are the Tool Artefacts.

### 4.2.1 The System Engineering Environment Descriptor (Output of the Platform Builder Process)

The diagram presented in the Figure 4.12 goes deeper into the SEE configuration schema, presenting the structure of each element of the two branches and their respective attributes. This diagram is called the SEE Descriptor and is the output of the Platform Builder process:

---

\(^{12}\) In order to specify the peer, it is only required to have the link between Tool and IOS Service. Complementarily, it is also required to know the respective Activity and Engineering Method so that it is possible to know why they are being used e.g. for tracking purposes.

\(^{13}\) Looking at Figure 4.7, it is possible to see that a SW Component Descriptor (i.e. a Tool Descriptor) has a relationship to SW Functions. Those SW Functions are the ETFs. This makes implicitly the link between the Tool/Brick and the Engineering Tool Function (ETF) blocks in Figure 4.10 and therefore that is the reason why there is no explicit link between these two blocks on the discussed diagram.
It is possible to see that most of the elements have an "id" attribute, some have also a "name" attribute and, in the case of the SEE itself, it still has further attributes as the "epf-process-id" and "epf-process-desc" ones. Here it is also possible to see the mapping of the process branch to different Platform Builder Meta-Model / SPEM elements. It is also important to notice that:

- The SEE configurator ensures that the selected tools for an EM provide the ETFs that are used in such EM;
- The tool which provides a given ETF is specified when storing the IOS Tool Chain Link Peer information.

Finally, Figure 4.13 presents the complementary SEE descriptor which specifies the Process and Tool Artefacts:
Process and Tool Artefacts have the same attributes but have the following differences:

- Process Artefacts are generic ones (not related to a tool) e.g. a system model;
- Tool Artefacts are provided by a tool e.g. a rhapsody internal block diagram.

### 4.2.2 The relationship between the different descriptors and the SEE

This section presents the main different descriptors or elements used in the Platform Builder process, which were already presented in the diverse developed meta-model diagrams, and the way they are related to the SEE descriptor as well as to each other. Figure 4.14 shows this relationship:

![Figure 4.14: Relation between different descriptors and artefacts](image-url)
5 Platform Builder Catalogues and Data Models

In this chapter, information about further Platform Builder-related elements are given in order to clarify to the reader the relationship between elements from the Platform Builder Meta-Model and other implementation-related components used to implement the Platform Builder itself. These elements were essentially conceived using the EPF Composer (see References) and therefore it is recommended further understanding about it.

5.1 The Platform Builder Method Contents Plug-in

Considering that EPF Composer is an authoring tool that provides User Interfaces which enables the formalization of the development process based on the SPEM Meta-Model then it was used for describing SEE-related processes.

In the tailoring phase all steps for authoring the process is done using the functionality of EPF Composer and therefore it is called the Platform Builder Method Contents Plug-in (PB-MCP) which is used to describe different useful catalogues. These catalogues compose the CRYSTAL Reference Library.

One of the main goals of the tailoring phase is to obtain a list of EMs which shall address the tool selection task for the configuration phase. Some part of the tailoring process is based and guided through two basic elements:

- **PB-MCP** used to create the different catalogues during the tailoring phase;
- **PB-MCP Methodology** which explains how to use the PB-MCP elements in a proper way when tailoring a process in order to obtain the required catalogues.

In this context, a Tailored Process is an instance of the first discussed PB-MCP Template, which can be seen in detail in Figure 5.1:

```
PlatformBuilder- Tailored Process instances
   |-- PB-MCP - instance
       |-- Activity Catalogue
       |-- EM Catalogue
       |-- Disciplines
          |-- platform_builder_disciplines
             |-- activity_discipline
             |-- em_discipline
```

Figure 5.1: PB Method Content Template

The PB-MCP template has the structure depicted in Figure 5.1: it contains the catalogues for Process Activities and Engineering Methods which are relevant to the specified Development Process as it will be more extensively described in the next sections. These last quoted catalogues shall also be used to define the Tailored Process.
5.2 The CRYSTAL Reference Library

As mentioned before, the CRYSTAL Reference Library is a collection of different catalogues that are used within the Platform Builder Process and that group, in an organized way, all the required information that is necessary to accomplish its objective e.g. to be able to have in the output an validated SEE. These catalogues can be seen in Table 5.1:

<table>
<thead>
<tr>
<th>Data Object name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activities Catalogue</td>
<td>It is a container that collects in a formal way all identified Activities within an engineering domain.</td>
</tr>
<tr>
<td>Engineering Methods</td>
<td>It is a container that collects in a formal way all identified Engineering Methods within an engineering domain.</td>
</tr>
<tr>
<td>Development Process</td>
<td>It is a Guidance element, under CRYSTAL reference Library, which helps to define the system development process.</td>
</tr>
</tbody>
</table>

Table 5.1: The CRYSTAL Reference Library or the Platform Builder Catalogues

Figure 5.2 presents the real structure of this library, which is used as input for the Platform Builder Modeller:

CORE--- base_domain
   |-- roles
   |-- work products {artefacts}
   |-- tasks {process activities}

Practices --- base_domain
   |-- tasks {Engineering Methods}

Process --- base_domain
   |-- Delivery Process {development process}

Guidances –
   |--base-PB – {guide for PB usage}
   |-- base-Modeller - {guide to Tailor the process}
   |-- PB-MCP - Template - {Template for PB methodology}

PlatformBuilder- Tailored Process instances
   |-- PB-MCP - instance

Figure 5.2: Example of the CRYSTAL Reference Library

The CRYSTAL Reference Library is also a facility to formally describe a development process and provide guidance and templates in order to allow the easy organization of the information within the Platform Builder approach. It is composed of different packages that group Method Content elements as it can be seen in Table 5.2:
<table>
<thead>
<tr>
<th>Package</th>
<th>Description</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORE</td>
<td>It contains Activities, Roles and Work Products, formalized for different domains, which can be imported in order to compose a Development Process.</td>
<td>For each identified domain there are: Roles, Work Products and Tasks. Here work products define Artefacts and tasks define process activities.</td>
</tr>
<tr>
<td>Practices</td>
<td>It contains formalized generic Engineering Methods to be applied within the Tailored Process phase.</td>
<td>For each identified domain, tasks define Engineering Methods.</td>
</tr>
<tr>
<td>Process</td>
<td>It represents the Development Process (based on the SPEM meta-model) and it can be used by a Process Engineer as guidance in the definition of the Tailored Process activities and their sequence.</td>
<td>For each domain, a delivery process defines a Development Process.</td>
</tr>
<tr>
<td>Guidance</td>
<td>It groups user manuals which are navigable by the end user e.g. PB guide and tailored Process guide. It includes also the PB-MCP Template that is the baseline for creating new specific Tailored Processes.</td>
<td>It contains the template for the PB methodology: the PB-MCP. By way of example, an instance of this template can be seen in Figure 5.1. It contains the Activity Catalogue and the EM Catalogue. These catalogues import Activities and Engineering Methods from CORE and Practices packages respectively.</td>
</tr>
<tr>
<td>Platform Builder</td>
<td>It contains the instantiated Tailored Process (from the template and specified for a specific domain use case development process).</td>
<td>It contains all instantiations of a Tailored Process for a specific Use Cases.</td>
</tr>
</tbody>
</table>

Table 5.2: The packages that group the Method Content elements within CRYSTAL Reference Library

Here it is important to notice that the Development Process Guidance is composed by CORE and Process contents.

The different catalogues that compose the CRYSTAL Reference Library are described in the next subsections.

### 5.2.1 Activities Catalogue

Activities are identified within a development process as tasks to be performed during the system development and they are formalized in the Activities Catalogue.

### 5.2.2 Engineering Methods Catalogue

Engineering Methods are identified within a development process for a specific Use Case and they represent the adopted methods to accomplish process activities. They can be considered as practices which are used in a specific domain for a specific Use case. Engineering Methods are categorized and formalized in the EMs Catalogue.

### 5.2.3 Development Process (Guidance)

In general, a Development Process is defined to formalize activities and their sequence and to define relevant information like roles and work products in each business domain. The Development Process definition might have diverse instances depending on different Use cases. A Development Process is used as Guidance for tailoring a process in the Platform Builder approach; indeed, during the tailoring phase, a Development Process can be used as a Guideline to describe the tailored process for a specific Use Case.
5.3 Catalogues for Tool Description

The last category of catalogues which are part of the Platform Builder approach are the Tool Description Catalogues. These ones are defined externally to the CRYSTAL library and are defined using xml files. The items presented in Table 5.3 compose this class of catalogues:

<table>
<thead>
<tr>
<th>Data Object name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engineering Tool Functions Catalogue</td>
<td>It is a container of Engineering Tool Functions identified in the scope of project CRYSTAL.</td>
</tr>
<tr>
<td>Tools Catalogue</td>
<td>It is a container of Tool Bricks descriptors, defined on the basis of provided ETFs.</td>
</tr>
</tbody>
</table>

Table 5.3: The CRYSTAL Reference Library or the Platform Builder Catalogues

These catalogues are presented in the next sections.

5.3.1 Engineering Tool Function Catalogue

The Engineering Tool Function Catalogue contains Engineering Tool Function descriptions that are used for two different purposes within the Platform Builder:

- It is used by Tool Vendors when creating the Tool Descriptor of a given Tool in order to identify the ETFs this tool supports;
- It is used in the Configuration phase of the Platform Builder workflow for identifying which ETF will be used in each EM that is part of the process.

Therefore, this catalogue lists the agreed set of ETF that can be used for expressing, on one hand, the functionality that is required in an EM and, on the other hand, the features provided by a Tool. Using this common set of information for expressing process required needs, tool features and input and output artefacts, it supports later on the searching of the tools that can be used in each step of an EM.

Moreover, this catalogue also defines the interaction between the ETFs. This is used for specifying which are the valid pairs of IOS services (provider and consumer). An example of the ETF Descriptor contained within the ETF Catalogue can be seen in Figure 5.3:

```xml
<ef of-id="IOS_OSLC_RM_2.0_CON_GET_REQ">
  <ef-name>Consume Requirement List - OSLC 2.0</ef-name>
  <ef-type>IOS</ef-type>
  <ef-ios-operation>Read</ef-ios-operation>
  <ef-description>Obtain a list of requirements based on the OSLC specification based on a search</ef-description>
  <ef-version>0.1</ef-version>
  <ef-ios-spec>http://open-services.net/ns/relm#</ef-ios-spec>
  <ef-engineering-domain>Requirement Management</ef-engineering-domain>
  <ef-tool-class>Any</ef-tool-class>
  <ef-artifacts>
    <ef-artifact of-artifact-id="IOS_OSLC_RM_2.0_Requirement" ef-artifact-mode="output">
      <ef-artifact-name>Requirement</ef-artifact-name>
      <ef-artifact-description>List of requirements satisfying the search criteria.</ef-artifact-description>
      <ef-artifact-format-code>http://open-services.net/ns/relm#Requirement</ef-artifact-format-code>
    </ef-artifact>
  </ef-artifacts>
</ef>
```

Figure 5.3: Example of ETF descriptor contained in the ETF catalogue

5.3.2 Tools Catalogue

The provisioning of a detailed specification in order to allow tool providers to describe their tools in a standardized way is also foreseen within Platform Builder scope. The motivation of having such specification is to support in the PB the search, filtering and selection of the most suitable tools for a specific Tailored
Process, according to different criteria; being the most important one the ETF required in each Engineering Method of the Tailored Process (comprising both internal ETF and IOS services). For this reason it has been defined the Tool Descriptor schema as a standard way to describe tools that can be used within the SEE. Then, tool vendors can define their tools using such specification and generating a Tool Descriptor for each tool. Finally, Tool Descriptors are grouped in Tool Catalogues.

Platform Builder makes use of two different Tool Catalogues: RTP Tools Catalogue and Available Tools Catalogue. The RTP Tools Catalogue contains the Tool Descriptors of the Available Tools in the market that support the IOS while the Company Available Tools Catalogue contains the Tool Descriptors of the tools that are available for use within a given company and which provide IOS support. The first one is used during the configuring phase of the PB workflow for selecting the tools that will be used for supporting the ETF that are needed in a given EM while the second is used during the validation phase for checking which tools are really available in the company.

The associations between the different catalogues are presented in Figure 5.4.

![Figure 5.4: Relation between the different tool catalogues](image)

An example of the Tool Descriptor contained within the Tool Catalogue can be seen in Figure 5.5:

```xml
<sv-component-descriptor id="RQS-2014.1" name="Requirements Quality Suite" version="2014.1" vendor="Reuse"> 
<description>The goal of the Requirements Quality Suite (RQS) is to efficiently manage the quality of both, requirements and knowledge as real assets for the organisation.</description> 
</sv-component-descriptor>

<extended-by sv-component-id="RQS-2014.1-IOS-RM-CON-Adapter"></extended-by>

<!-- 1 or more. IDs obtained from the RF ontology -->
<sv-function id="ANALYSIS_REQUIREMENTS" type="TOOL"/>

<!-- This is not very important for the first prototype. But it stills need a lot of work. -->
<it-requirements>
    <network-deployment></network-deployment>
</it-requirements>

</sv-component-descriptor>

<sv-component-descriptor id="RQS-2014.1-IOS-RM-CON-Adapter" name="RQS IOS RM Adapter" version="2014.1" vendor="Reuse"> 
<description>IOS Requirement Manager Adapter for RQS.</description> 
</sv-component-descriptor>

<extended sv-component-id="RQS-2014.1"></extended>

<!-- 1 or more. IDs obtained from the RF ontology -->
<sv-function id="IOS_OSGIC_RM_2.0_CON_GST_REQ" type="IOS"/>
<sv-function id="IOS_OSGIC_RM_2.0_CON_UPD_REQ" type="IOS"/>
<sv-function id="IOS_OSGIC_RM_2.0_CON_DIS_REQ" type="IOS"/>
```

![Figure 5.5: Example of Tool descriptor contained in the Tool catalogue](image)
## 6 Glossary

These glossary entries are a subset of the central Crystal glossary at https://projects.avl.com/11/0154/Lists/Glossary/AllItems.aspx.

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
<th>Term/ description inherited from</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity</td>
<td>Activity is a concrete work identified within a Development Process and it represents a general unit of work assignable to specific user able to perform it i.e. 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. An activity is detailed using Engineering Functions.</td>
<td>Partially from WP 602 and CRYSTAL Glossary</td>
</tr>
<tr>
<td>Artefact</td>
<td>A physical piece of information that is used or produced by an engineering tool. Its semantics and syntax is defined by standard formats or by proprietary formats.</td>
<td>CRYSTAL Glossary</td>
</tr>
<tr>
<td>Available Tools</td>
<td>Available Tools are the available tools in the company which are or not linked by the IOS.</td>
<td></td>
</tr>
<tr>
<td>Business Process or Development Process</td>
<td>It is the system life cycle development and management process that represents a whole given process, covering development phase and product management phase. Life cycle management comprises tasks of product management process as: traceability, configuration and versioning, user and process management. In the other hand, life cycle development covers tasks of development process (as requirements elicitation and formalization, design, implementation, validation and verification).</td>
<td></td>
</tr>
<tr>
<td>System Development Process</td>
<td>A process where specified activities are relevant to a development of a defined system.</td>
<td>CRYSTAL Glossary</td>
</tr>
<tr>
<td>RTP Tools Catalogue</td>
<td>RTP Tools Catalogue contains all the Tools adapted to work within the IOS in CRYSTAL RTP.</td>
<td></td>
</tr>
<tr>
<td>Engineering Tool Function (ETF)</td>
<td>Engineering Tool Function is a detailed function of an Activity that refers to an action performed by means of a Tool Brick. ETFs can be of two types: IOS Service or Internal ETF.</td>
<td>CRYSTAL Glossary</td>
</tr>
<tr>
<td>Engineering IOS Function</td>
<td>Engineering IOS Function is an engineering tool function that contributes to interoperability aspects between different tools.</td>
<td></td>
</tr>
<tr>
<td>Engineering Method (EM)</td>
<td>It describes a method that accomplishes an identified activity defined by steps to be performed and artefacts to be used, created or edited i.e. it defines HOW to satisfy Process Activities.</td>
<td></td>
</tr>
<tr>
<td>Internal ETF</td>
<td>A functionality provided by a single Tool Brick and that performs operations within the Tool Brick itself.</td>
<td>CRYSTAL Glossary</td>
</tr>
<tr>
<td><strong>IOS Service</strong></td>
<td>It is an engineering function that is provided by a Tool Brick and that supports interoperability between different Tools.</td>
<td>CRYSTAL Glossary</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td><strong>IT Infrastructure</strong></td>
<td>IT infrastructure is a combined set of hardware, software, networks and services/facilities where a tool-chain is deployed and needed network is instantiated, services are made available, and security and safety constraints are applied. There are three different IT infrastructure descriptions: IT infrastructure as required by the process, available IT infrastructure in the company and validated IT infrastructure.</td>
<td>Partially from WP 602 Glossary</td>
</tr>
<tr>
<td><strong>Plugged-in Tool-Chain (Implemented Tool Set)</strong></td>
<td>It is the instantiation of a tool-chain within the available IT infrastructure.</td>
<td></td>
</tr>
<tr>
<td><strong>Process</strong></td>
<td>It is a combination of activities to be performed in order to accomplish a goal for a specific Use Case. The activities may be in sequence, iterative or in parallel. In other words, it is a set of interrelated or interacting activities which transforms inputs into outputs.</td>
<td>Partially from CRYSTAL Glossary / ISO/IEC 15288</td>
</tr>
<tr>
<td><strong>Process Activity</strong></td>
<td>It defines an activity to be performed which produces tangible work products. It defines WHAT is done in the process.</td>
<td></td>
</tr>
<tr>
<td><strong>Project/Process Data</strong></td>
<td>Project Data are relevant data to the project which is necessary to setup up the SEE. Project Data are Work Products and Roles defined in the Tailored Process. Roles will be assigned to Users identified within a company organization which (Users that play a Role) uses the SEE.</td>
<td>Partially from WP 602 Glossary</td>
</tr>
<tr>
<td><strong>Role/Role Use</strong></td>
<td>A Role Use represents a Role in the context of one specific Activity. Every breakdown structure can define different relationships of Role Uses to Task Uses and Work Product Uses. Therefore, one role can be represented by many Role Uses, each within the context of an Activity with its own set of relationships. Every Role Use can reference only one Role Definition. However, a Role Definition can be represented by many Role Uses. A Role Use can select a sub-set of valid Qualifications defined for the Role Definition for this one use of the Role Definition in the context of a particular Activity. It defines a group of skills with a typical function within an organisation.</td>
<td>Partially from SPEM 2.0</td>
</tr>
<tr>
<td><strong>System Engineering Environment (SEE) / SEE configuration</strong></td>
<td>A framework where a Tool Chain is instantiated within the organization IT infrastructure, to perform a system development process (specified following SPEM). This framework is set up using the project data. SEE in the Platform Builder has to be configured and validated. In other words, it is the development environment consisting of Tool Bricks, IT information and Project Data to be used by Users to develop a system.</td>
<td>Partially from CRYSTAL Glossary</td>
</tr>
<tr>
<td><strong>Tailored Process/ Tailored Project Process</strong></td>
<td>Tailored Process has to define all aspects relevant to a specific Use-Case and it defines in detail the workflow in terms of tasks to be performed and their sequence, work products to be produced, roles that perform tasks and tools to be used to perform tasks. Tailored process represents an input of the Platform Builder workflow.</td>
<td></td>
</tr>
<tr>
<td><strong>Tool Brick</strong></td>
<td>A software application or component, described in terms of the offered functions, meant to support Engineering Methods adopted in a system development process.</td>
<td>CRYSTAL Glossary</td>
</tr>
<tr>
<td>Tool Chain</td>
<td>A complete set of interoperable Tool Bricks meant to support Engineering Methods adopted in a system development process.</td>
<td>CRystal Glossary</td>
</tr>
<tr>
<td>----------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Use Case</td>
<td>It is a CRystal Domain Use Case and describes information concerning the development of a system in a given company.</td>
<td></td>
</tr>
<tr>
<td>User</td>
<td>It is the person that plays a Role and which shall access and use the deployed SEE or an individual who or group that benefits from a system during its utilization. The User definition implies to have access to general information of a person and user management information to control access rights.</td>
<td>Partially from CRystal Glossary / ISO/IEC 15288</td>
</tr>
<tr>
<td>Work Product</td>
<td>Work Products are in most cases tangible work products consumed, produced, or modified by Tasks. They may serve as a basis for defining reusable assets. Roles use Work Products to perform Tasks and produce Work Products in the course of performing Tasks. Work Products are the responsibility of Role Definitions, making responsibility easy to identify and understand, and promoting the idea that every piece of information produced in the method requires the appropriate set of skills. Even though one Role Definition might “own” a specific type of Work Product, other roles can still use the Work Product for their work, and perhaps even update them if the Role Definition instance has been given permission to do so.</td>
<td>SPEM 2.0</td>
</tr>
</tbody>
</table>

Table 6.1: Glossary Terms and Definitions
## 7 References

<table>
<thead>
<tr>
<th>[Author, Year]</th>
<th>Authors; Title; Publication data (document reference)</th>
</tr>
</thead>
</table>
8 Annex 1 - Compliance of CRYSTAL PB MM elements to SPEM

For contextualization purposes, it is presented here a preliminary mapping between SPEM elements and identified CRYSTAL Platform Builder Meta-Model elements which was done in the beginning of the project as a way to understand the gap between SPEM specification and required elements for the developed meta-model. Therefore, the main objective of this mapping was to find out if the elements needed for the CRYSTAL Platform Builder Meta-Model were covered (even partially) by the elements present in SPEM or if a further enrichment and expansion of SPEM was needed in order to cover all the missing aspects of the Platform Builder needs.

Consequently, this Annex has to be seen as a record of the evolution of the gap analysis executed in the early phases of the conception of the CRYSTAL Platform Builder Meta-Model, rather than complementary content for understanding the developed meta-model.

The next sections present the performed mapping both in relation to the Process and Content packages.

8.1 Mapping concerning process aspects (Process Structure package)

A preliminary mapping and gap analysis was performed concerning the process point of view. The basic employed approach in the execution of the mapping was to study the existing elements that compose the SPEM Process Structure package and then try to fit the CRYSTAL Platform Builder Meta-Model elements to them. Figure 8.1 shows the original studied package structure:

![Figure 8.1: Taxonomy of classes defined in the Process Structure package (source: SPEM 2.0)](image)

The performed mapping can be seen in Figure 8.2.
Figure 8.2: Mapping of CRYSTAL Platform Builder Meta-Model elements to the Process Structure package

The following short descriptions that come from SPEM 2.0 specification support the choices of such mapping:

- **Activity**: “An Activity is a Work Breakdown Element and Work Definition that defines basic units of work within a Process as well as a Process itself. In other words, every Activity represents a Process in SPEM 2.0.”

- **Process Parameters**: “A Process Parameter is a Work Definition Parameter and Breakdown Element that is used for process definitions. It defines input and output meta-types to be Work Product Uses.”
  - Artefact Relationship, IT Requirements and Permission were mapped to Process Parameter in SPEM Process Structure package.

- **Role Use**: “A Role Use is a special Breakdown Element that either represents a performer of an Activity or a participant of the Activity. If it is a performer, the Role Use and Activity need to be related via a Process Performer. If it is a participant, then the Role Use is stored in the nestedBreakdownElement composition of the Activity and might be used by one of the sub-activities as a performer and/or a Process Responsibility Assignment. Role Uses are only valid within and specific to the context of an Activity. They are not to be reused across activities.”
  - Role was mapped to Role Use in SPEM Process Structure package.

- **Work Product Use**: “A Work Product Use is a special Breakdown Element that either represents an input and/or output type for an Activity or represents a general participant of the Activity. If it is an input/output, then the Work Product Use needs to be related to the Activity via the Process Parameter class.”
8.2 Mapping concerning methods aspects (Method Content package)

Following the same idea from the previous section, a preliminary mapping and gap analysis was performed concerning this time the method point of view. Again, the employed approach consisted in executing the mapping by starting the examination of the existing elements that compose the SPEM Method Content package and then trying to fit the CRYSTAL Platform Builder Meta-Model elements to them. Figure 8.3 shows the original studied package structure:

![Taxonomy of Core Describable Elements](source:SPEM 2.0)

The resultant mapping can be seen in Figure 8.4.

![Mapping of CRYSTAL and SPEM MMs concerning the Method Content package](source:SPEM 2.0)

The following short descriptions that come from SPEM 2.0 specification support once again the choices of such mapping:

- **Role Definition**: “A Role Definition is a Method Content Element that defines a set of related skills, competencies and responsibilities. Roles are used by Task Definitions to define who performs them as well as to define a set of Work Product Definitions they are responsible for.”
  - Artefact Relationship, IT Requirements, Permission and Role were mapped to Role Definition in SPEM Method Content package.
- **Step**: “A Step is a Section and Work Definition that is used to organize a Task Definition’s Content Description into parts or subunits of work. Steps inherit the subSection decomposition from Section and can therefore describe sub-Steps nested into Steps.”
  - *Engineering Method, Engineering Tool Function, Internal ETF and IOS Service* were mapped to Step in SPEM Method Content package.

- **Task Definition**: “A Task Definition is a Method Content Element and a Work Definition that defines work being performed by Roles Definition instances. A Task is associated to input and output Work Products.”

- **Tool Definition**: “A Tool Definition is a special Method Content Element that can be used to specify a tool's participation in a Task Definition.”
  - *RTP Component, SW Function and Tool Brick* were mapped to Tool Definition in SPEM Method Content package.

- **Work Definition**: “Work Definition is an abstract Classifier that generalizes all definitions of work within SPEM 2.0. Work Definition defines some default associations to Work Definition Parameter and Constraint. Work Definitions can contain sets of pre and post-conditions defining constraints that need to be valid before the described work can begin or before it can be declared as finished. Note that general UML constraints inherited via Classifier can be used to define additional constraints and rules for Work Definitions.”
  - *Process, Process Activity, Tailored Process and Use Case* were mapped to Work Definition in SPEM Method Content package.

- **Work Product Definition**: “Work Product Definition is Method Content Element that is used, modified, and produced by Task Definitions. Work Product Definitions can be related to other Work Product Definitions via the Work Product Definition Relationship.”
  - *Activities Catalogue, Artefact, EM Catalogue, ETF Catalogue, IOS Tool Chain Link, IOS Tool Chain Link Peer, Machine, Network, SEE, SW Component Descriptor, Tool Catalogue, Tool Chain and User* were mapped to Work Product Definition in SPEM Method Content package.