



Defence Research and  
Development Canada

Recherche et développement  
pour la défense Canada



# Capability Engineering Process Documentation Framework

*How to document a CEP within Aladin*

Christophe Nécaille  
Claire Lalancette  
DRDC Valcartier

**Defence R&D Canada – Valcartier**

Technical Memorandum  
DRDC Valcartier TM 2013-176  
February 2013

Canada





# **Capability Engineering Process Documentation Framework**

*How to document a CEP within Aladin*

Christophe Nécaille  
Claire Lalancette  
DRDC Valcartier

## **Defence R&D Canada – Valcartier**

Technical Memorandum  
DRDC Valcartier TM 2013-176  
February 2013

- © Her Majesty the Queen in Right of Canada, as represented by the Minister of National Defence, 2013
- © Sa Majesté la Reine (en droit du Canada), telle que représentée par le ministre de la Défense nationale, 2013

## Abstract

---

A formal process documentation framework is a mandatory asset for the PACEM Project (Patterns and Agility for Capability Engineering Methodology-Requirement Engineering), to enable the use of process patterns. This report details the Capability Engineering Process Documentation Framework established during the PACEM-RE Project. It introduces various background elements and presents the components and structure of the process documentation through a process engineering repository.

## Résumé

---

Le projet PACEM-RE (Pattern et agilité pour les méthodes d'ingénierie de capacité) nécessitait l'établissement d'un cadre plus formel de documentation pour la publication des processus d'ingénierie, afin de pouvoir utiliser le concept de *process pattern*. Ce rapport détaille le cadre de documentation des processus d'ingénierie de capacité établi durant ce projet. Il introduit différentes notions de bases et présente les composantes et la structure du cadre de documentation par le biais d'un entrepôt de processus d'ingénierie.

This page intentionally left blank.

## Executive summary

---

### **Capability Engineering Process Documentation Framework: How to document a CEP within Aladin**

**Christophe Nécaille; C. Lalancette; DRDC Valcartier TM 2013-176; Defence  
R&D Canada – Valcartier; February 2013.**

The PACEM-RE Project (Patterns and Agility for Capability Engineering Methodology-Requirement Engineering) aims to validate the efficiency of the process patterns concept to augment agility for capability engineering methodology. A process design and publication tool was developed to effectively introduce the concept of and to document some of these process patterns. The tool, known as Aladin, was implemented by adapting an open-source project called the Eclipse Process Framework (EPF).

The documentation framework detailed in this report uses the concept of Agility, Process Patterns, as well as the principles of the Method Engineering domain. It includes all the elements of process engineering either static (deliverables, tools, roles ...) and dynamic (processes, patterns, ...) ones. A set of rules and principles assist the process engineer in the adaptation of an engineering process using process patterns. It guides the user in the process elaboration, documentation and publication.

With the use of this framework, the Capability Engineering Process can be adapted to better suit the particularity of the specific engineering process it supports.

## Sommaire

---

### Capability Engineering Process Documentation Framework: How to document a CEP within Aladin

**Christophe Nécaille; C. Lalancette ; DRDC Valcartier TM 2013-176; R & D pour la défense Canada – Valcartier; février 2013.**

Le projet PACEM-RE (Pattern et agilité pour les méthodes d'ingénierie de capacité) vise à établir l'efficacité du concept de patrons de processus (*process patterns*) pour augmenter l'agilité des processus d'ingénierie de capacité. Un outil a été développé pour pouvoir mettre en oeuvre les patrons de processus. Basé sur le code open-source EPF (Eclipse Process Framework) le logiciel Aladin permet de documenter et de publier les processus ainsi que les patrons de processus.

Le cadrage de documentation de processus exposé dans ce rapport utilise les concepts d'agilité, de patrons de processus ainsi que les principes de domaine de l'ingénierie de méthodes. Il inclut l'ensemble des éléments du domaine de l'ingénierie de méthodes, qu'ils soient statiques (livrables, outils, acteurs, etc.) ou dynamiques (processus, patrons de processus, etc.) Un ensemble de règles assiste le concepteur de processus d'ingénierie de capacité dans l'utilisation de patrons pour adapter le dit processus et permettre sa documentation, sa publication et sa mise en œuvre.

Le processus d'ingénierie de capacité peut donc, grâce à l'utilisation d'Aladin et du cadrage de documentation détaillé dans ce rapport, être adapté afin de répondre aux caractéristiques particulières du projet d'ingénierie spécifique qu'il supporte.

# Table of contents

---

Abstract .....	i
Résumé .....	i
Executive summary .....	iii
Sommaire .....	iv
Table of contents .....	v
List of figures .....	vii
List of tables .....	viii
Acknowledgements .....	ix
1 Introduction.....	1
2 Documentation framework foundations .....	3
2.1 Method engineering basics .....	3
2.2 EPF, Aladin, SPEM and UMA.....	6
2.3 Agility, patterns and process patterns.....	6
2.3.1 Thoughts on agility .....	6
2.3.2 Patterns and process patterns .....	7
2.3.3 Patterns .....	8
2.3.4 Process patterns .....	8
3 Framework structure.....	10
3.1 General principles.....	10
3.2 The CEP library structure.....	12
3.2.1 Core.common plug-in .....	12
3.2.1.1 Content packages .....	13
3.2.1.2 Standard categories .....	14
3.2.1.3 Custom categories.....	15
3.2.2 Pattern plug-ins.....	15
3.2.2.1 Method content .....	15
3.2.2.2 Process content.....	16
4 Framework usage.....	18
4.1 Creation of a generic CEP .....	18
4.1.1 Completion sequence.....	18
4.1.2 General rules.....	20
4.1.3 Method elements completion rules .....	21
4.1.3.1 Guidance .....	23
4.1.3.2 Work products.....	25
4.1.3.3 Roles .....	26
4.1.3.4 Task.....	27
4.1.4 Activity completion rules .....	29

4.1.5	Stage/iteration rules .....	32
4.1.6	Publication rules .....	36
4.1.7	Validation checklist .....	38
4.2	CEP adaptation .....	42
4.2.1	Basic structure creation.....	42
4.2.2	Pattern description .....	43
4.2.2.1	General information .....	43
4.2.2.2	Context information .....	44
4.2.2.3	Solution information .....	45
4.2.3	Completion sequence.....	47
4.2.4	General process pattern rules .....	49
4.2.5	Method elements rules.....	49
4.2.6	Activity/iteration/stage/delivery process rules.....	51
4.2.7	Validation checklist .....	52
4.2.8	Publication .....	52
5	Conclusion.....	53
Annex A	Unified Method Architecture Metamodel .....	57
Annex B	List of acceptable verbs .....	59
Annex C	Step-by-step creation of the high-spin pattern.....	61



## List of figures

---

Figure 1: The Capability Engineering Process .....	1
Figure 2: Method Engineering Metamodel .....	5
Figure 3: Pattern as a Triplet .....	8
Figure 4: Method Library Structure.....	10
Figure 5: Examples of A Method Library Structure.....	11
Figure 6: CEP High-Level Method Library Structure .....	12
Figure 7: CEP Custom Categories.....	15
Figure 8: Editor for Method Content Element.....	22
Figure 9: Description Tab for Activity Edition .....	30
Figure 10: Work Breakdown Structure Tab for Activity Editing.....	31
Figure 11: CEP Iterations and Stages .....	32
Figure 12: Description of an Iteration .....	33
Figure 13: Generated (A) and Reviewed (B) Workflows for an Iteration.....	35
Figure 14: Publishing Options .....	38
Figure 15: Process Pattern Location.....	42
Figure 16: Process Pattern Template – General Information Section .....	43
Figure 17: The Context Information Section.....	45
Figure 18: The Solution Information Section.....	47
Figure 19: UMA content model .....	57
Figure 20: UMA Process Elements .....	58
Figure 21: Process Pattern Documentation .....	62
Figure 22: Method Configuration.....	62
Figure 23: CET Lead Assistant Role Creation .....	63
Figure 24: Sprint Notebook Artifact Creation.....	64
Figure 25: Task Modification Using the Contributes Variability Type.....	65
Figure 26: Process Structure Modification.....	69
Figure 27: Activity diagram of the HS_Sprint stage .....	70
Figure 28: Package Selection .....	71

## List of tables

---

Table 1: Description of Content Packages .....	13
Table 2: Standard Categories Organization.....	14
Table 3: Process Packages Description .....	16
Table 4: Description of Guidance Editor.....	23
Table 5: Description of Work Product Editor .....	25
Table 6: Description of Role Editor .....	26
Table 7: Description of Task Editor .....	28
Table 8: Modifications to Custom Categories for Navigation Menu Customization .....	36
Table 9: Information for Configuration Editor .....	37
Table 10: Summary of Mandatory Fields for Method Elements .....	40
Table 11: Variability Mechanisms .....	51
Table 12: List of Acceptable Verbs .....	59
Table 13: New tasks – MEE package .....	65
Table 14: Modifications to Existing Tasks – MEE Package .....	66
Table 15: Modified Tasks – ACG package .....	67
Table 16: Modified tasks – DOA package .....	67
Table 17: Modified tasks – AFDO package .....	68
Table 18: Modified tasks – DSOSA package .....	68

## Acknowledgements

---

The authors would like to thank their colleague, Geneviève Dussault, for her active participation in the structuring of the documentation framework. Her involvement made it possible to better understand and design the CEP documentation framework.

The authors would also like to acknowledge Jocelyn Leclerc from CGI Québec for his exploratory work on the library structures of EPF. Mustapha Zaraket, Jean-Pierre Bélanger (CGI Québec) and Mathieu Turcotte (Université Laval) must also be recognized for their involvement in the technical aspects of the Aladin

This page intentionally left blank

# 1 Introduction

Since 2002, the Department of National Defence and the Canadian Forces have been implementing a centralized capability-based planning approach as a core element of a new force development process.

As part of this initiative, a capability engineering process (CEP) was defined between 2003 and 2007. The CEP is initiated from a statement of capability gap/goal and is completed when a set of potential force development options and a recommendation for the best option in terms of performance, cost, schedule and risk is achieved. More details about the CEP can be found in [1] and [2].

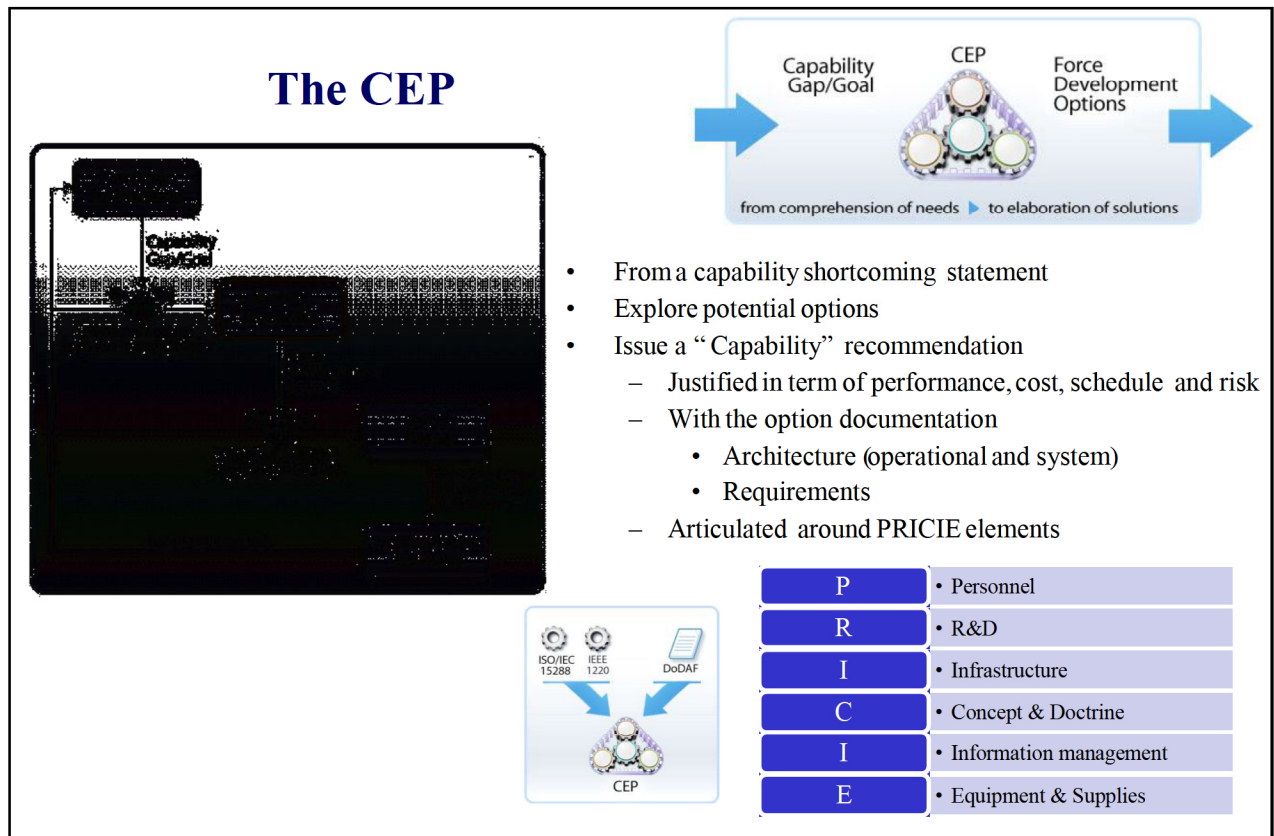


Figure 1: The Capability Engineering Process

## PACEM context

CEP was used within various projects at DRDC Valcartier, either as is or after being adapted to specific needs:

- The first project, Joint Fire Support (JFS), loosely followed the first two stages of the CEP before employing its own management procedures. The use of the CEP by the JFS Project was demonstrated to the TTCP TP-4 Panel on several occasions [3].

- The CEP approach was also used within the Ground-Based Air and Munitions Defence (GBAMD) initiative. It was presented to Australia's Defence Science and Technology Organization [4].
- An adapted version of the CEP was designed for the Counter Anti-Ship Supersonic Sea Skimming Missiles (CAS4M) project. Two process patterns were identified: the "Management Hammer" and the "As-Is First" pattern, which can be found in [5].

The growing use of this process in these contexts highlighted the need for process agility. Following the Collaborative Capability Definition, Engineering and Management Technology Demonstration Project (CapDEM TDP), a research project called PACEM (Patterns and Agility for Capability Engineering Methodology) [5] was launched to assess the value of process patterns as a means to improve the agility of such processes.

### **ALADIN documentation framework**

To assist the documentation and publication of engineering processes built from reusable sub-process chunks, a prototype called Aladin was built. This tool, based on the open source Eclipse Process Framework, provides a means for documenting engineering methodology elements, such as activities, roles and work products, and combines these elements into activities and reusable process patterns. Aladin stores and manages the methodological documentation within a repository. Delivery processes are built from selected activities and patterns within the repository and are published on a website.

### **Need for a documentation framework**

The Aladin software offers much flexibility in how to document and organize its repositories. Different philosophies have been tested during the project with varying levels of success. A logical path must be followed to assemble process patterns in order to preserve the reusability and overall maintainability of the method repository.

### **Intended audience – goal**

The current publication is directed to process engineers and methodologists who need to document engineering processes and eventually publish them. It establishes a framework for capability engineering process documentation and provides guidance to novice Aladin users. This report is also meant to be used as a reference document for more experienced users.

### **Structure of the report**

The next section introduces the concept of process patterns and their use in capability engineering. Elements of situational method engineering are exposed, which are the pillars of process pattern approaches. The choice of structuring components such as the metamodel of process documentation are also justified. Section 3 presents the structure of the method repository. It also introduces the various elements and explains how to combine them. Finally, Section 4 details the use of the framework. The rules and way to document the generic method elements are listed here, as are the steps and rules to follow to customize the CEP through the use of process patterns. For illustrative purposes, a complete step-by-step documentation example for a patterned CEP is found in Annex C.



## 2 Documentation framework foundations

---

The goal of the CEP documentation framework is to provide structure, syntax and semantic rules to enforce a rigorous and consistent documentation of capability engineering processes. The framework should enhance the quality and reusability of the documentation.

The framework takes its roots from different theories that are briefly introduced here. Methods engineering basics introduce the vocabulary and different concepts that are used to describe the framework. A discussion on patterns, process patterns and agility follows, as these concepts are the core of the PACEM project.

### 2.1 Method engineering basics

The discipline of method engineering was developed based on the observation of the gap between theoretical engineering methods and their application in real situations. Situational method engineering is a term proposed by Kumar and Welke in [7] to describe the methodology of developing, customizing and/or configuring a situation-specific method from parts of existing methods. A complete state of the art was published recently by Henderson-Sellers and Ralyté in [8].

To clarify the distinction between methods and methodology, one must go up one level of abstraction and model the situation at a “meta” level (a metamodel is a language to describe methodologies, which are models of a process). In the literature, different approaches and metamodels have been published to enable method engineering, such as OMG SPDM (Software Process Engineering ISO/IEC 24744) [9] and the ISO/IEC 24744 Standard “Metamodel for Development Methodologies” [10]. The goal of these frameworks is to provide tools and structure to allow reusability and modularity of existing method chunks.

Figure 2 adapted from [11] presents a summary of these notions and introduces the different objects used in this report:

- The lowest level, M0 Process Execution level, addresses the work actually done by engineers, developers, architects (in other words, the designers) within a project, or a CEP instance. The objects at stake at this level are the development run, the artifacts and persons acting on the project. We are in the situation realm as opposed to the upper levels that target the knowledge realm.
- The Level M1 (process model) is the playground of the method engineers. This level addresses the “way of working.” At this level, methods such as ITIL, PMBoK or CEP are under the microscope. The transformation of a method or process toward the lower level is called process enactment.
- At the Level M2 (process metalevel), the methodologist experiments and designs process patterns and methodologies that are generic method models. In this level, generic concepts and patterns (or method chunks) are manipulated and assembled to build customized methods or processes (M1 Level). This operation is called method generation.

- Finally, the higher level, M3 Model Representation, deals with models of Level M2 items. This level is for the ontologists. Models of metaprocesses are built in order to represent the M2 Level. For instance, UML profile templates of process patterns belong to this level, as does the CEP documentation framework.

The PACEM project uses this concept as follows:

- The CEP documentation framework and the process pattern templates (M3) are the rule sets and representation models used to document the CEP and its process patterns, which together constitute a process repository (M2).
- Following this framework, it is possible to choose, modify and assemble chunks of existing process patterns to design a specific process pattern (for instance, the Hi-Spin pattern for fast-paced projects with a reduced designer team). This work is instrumented by the Aladin tool and repository.
- This specific pattern is then used to establish a particular method, the CEP process for project XX (Level M1). Once documented, this method can be published on a website using the Aladin publication features. At this level, execution of the process (Level M0) can be monitored and managed.
- Finally, at the lowest level, designers, architects, clients and project managers are executing the various tasks as documented in the upper level.



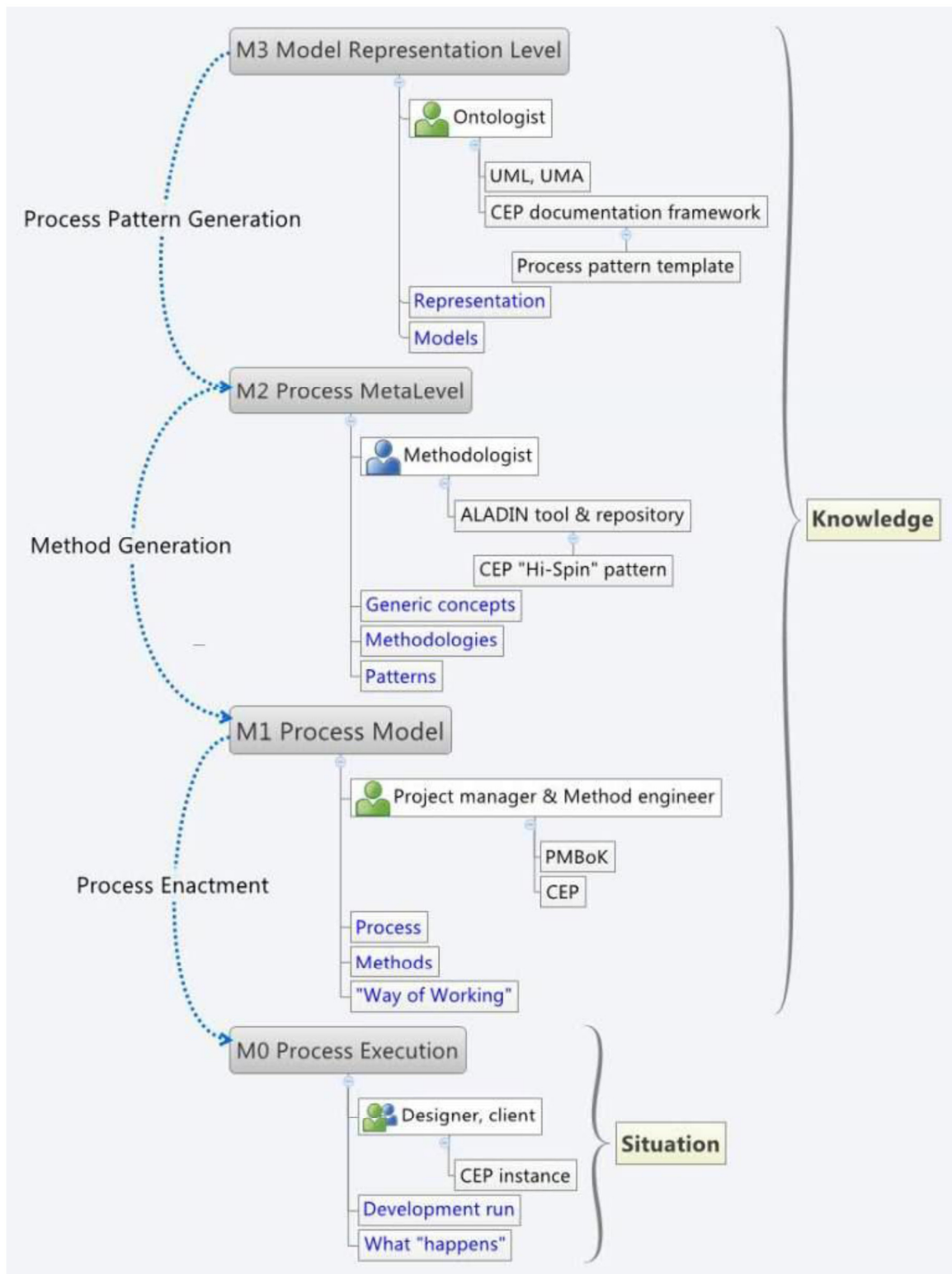


Figure 2: Method Engineering Metamodel

## **2.2 EPF, Aladin, SPEM and UMA**

Aladin is a Level M2 component of the PACEM project. Aladin is based on the open-source Eclipse Process Framework (EPF) software. EPF is an IBM-supported initiative [13]. Based on the Eclipse Rich Client Platform, it provides a framework in which to document a methodology; featuring both process and method content. Aladin is a modified EPF, in which some additional features have been added to better support the implementation of process patterns. The model elements for a process pattern have been added as have the corresponding user interfaces. A process selection feature has also been added to facilitate selection of applicable patterns based on a selection of preconditions.

Software Process Engineering Metamodel (SPEM) is a UML profile defined by the Object Management Group [9]. This UML profile has been under construction for some time. This level M3 metamodel has been the norm for representing software engineering processes. According to OMG, “The Software and Systems Process Engineering Meta-model (SPEM) is a process engineering meta-model as well as conceptual framework, which can provide the necessary concepts for modeling, documenting, presenting, managing, interchanging, and enacting development methods and processes. An implementation of this meta-model would be targeted at process engineers, project leads, project and program managers who are responsible for maintaining and implementing processes for their development organizations or individual projects.” The first version was officially released in 2002, while the current version, 2.0, was released in 2008.

UMA (Unified Method Architecture) is the metamodel used within EPF. This Level M3 metamodel was developed by IBM [14] and provides a high-level object-oriented architecture for documenting method and process content. Most of the UMA elements have been incorporated into the new SPEM 2.0 specification (from the OMG). UMA distinguishes the method content from the process content. Method content describes roles, the tasks that they perform, the work products produced by the tasks performed, and supporting guidance. They can be categorized into logical groups for indexing and display purposes. Method content elements are independent of a development life cycle. In fact, they are often reused in multiple development lifecycles. Processes describe the development life cycle. They define sequences of tasks performed by roles and work products produced over time. Processes are typically expressed as workflows or breakdown structures. The sequencing of the tasks within the breakdown structure usually represents different types of development lifecycles, such as waterfall, incremental and iterative. UMA has been used within Aladin for method representation. Annex A contains some UMA models.

## **2.3 Agility, patterns and process patterns**

### **2.3.1 Thoughts on agility**

The application scope of the CEP is constantly evolving. Designed to engineer strategic defence capabilities, it has been used by different research projects (such as JFS, GBAMD and CAS4M mentioned above) within DRDC to describe and engineer their systems of systems (SoS). The first project used the process during its inception stage, to establish the as-is situation of the

capability. A second one had to adapt it in order to deliver the as-is architecture description as soon as possible. The use of CEP by R&D projects confirmed its versatility.

Because of CEP validation activities, we were aware that engineering projects have to evolve in a dynamic context and environment. Funds, availability of stakeholders, classification of information and even customers' requirements evolve over the lifetime of the project and interact with its execution. A completely defined process from the start of a project is not a perfect solution and is a pitfall for adopting the CEP as it reinforces the perception of a rigid process. On the contrary, allowing flexibility in the process workflow during the project increases the risk of losing the rigour it provides. The demand for adapted processes started the reflection on possible means to provide agility to the CEP.

The review of different definitions of "agility" relevant to engineering processes led us to the following definition from [6]: "**Agility** is the **ability** to respond to **changing circumstances** where:

- **Ability** is characterized by readiness and speed of action
- **Response** is:
  - Making use of an existing configuration (by internal means); or
  - By reconfiguration (facilitated by external means).
- **Changing circumstances** may be:
  - A change in objective;
  - A change in environment; or
  - A change in condition.
- **Agility** is measured by:
  - Speed of action;
  - Cost in resource; and
  - Impact on effectiveness."

### 2.3.2 Patterns and process patterns

The concept of pattern and process patterns is used within the Aladin prototype, the PACEM process publication tool. A general presentation to pattern precedes the introduction of the process pattern concept. A short example is used to illustrate this concept. Later in the document, a complete section addresses the documentation process and rules to be followed to setup and publish a CEP using process patterns.

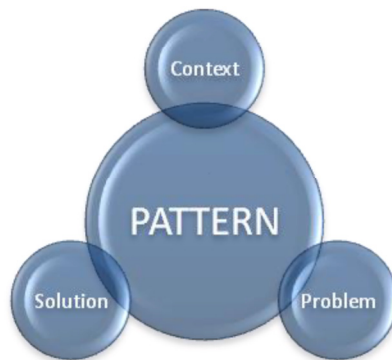
The PACEM project formulates the hypothesis that process patterns might be useful to increase the level of agility of an engineering process. Here, process patterns are reusable chunks of workflow, which can be reorganized in order to compose a new workflow, more suitable for the problem at stake.

The concept of pattern is borrowed from the work of Christopher Alexander [15], a civil architect who created a detailed description of how to use this concept to define a common set of rules for house building and civil architecture.

### 2.3.3 Patterns

In software engineering, this concept was popularized by Gamma, Helm, Johnson and Vlissides, also known as “the gang of four” [16]. Complete sets of patterns and a pattern language have been documented and used extensively by software developers, e.g., see [17] and [18]. To introduce the concept of pattern, Buschmann makes an analogy between the way experts work and patterns. In this study [19], the author shows that when exposed to a new problem to solve, experts will rarely apply a completely new solution. Instead, they recall in memory any problems with similar features that they have encountered in the past, analyze which part of the solution might be applicable, and design a specific solution based on these similarities.

A pattern is a tripartite relationship between a context, a problem and a solution (Figure 3). It allows collecting the essence of lessons learned and know-how from previous experiences.



*Figure 3: Pattern as a Triplet*

### 2.3.4 Process patterns

Process patterns are a collection of general techniques, actions and tasks from which an organization can develop a customized process that meets its exact needs. A process pattern is a pattern in which the solution component is a process itself. Therefore, a process pattern will document a context (what characterizes the situation where the pattern can be applied); a problem (what does the process want to solve) and a process as a solution (what is the generic organization of roles, tasks and work products recommended to evolve the situation).

In the literature, process patterns have been established and described in [17] and [18]. These patterns were intended for the software engineering domain. However, more recently, different authors have examined the use of this concept to systems and systems engineering as described in [12], [20] and [21]. The PACEM project hypothesis claims that process patterns can be useful to augment the agility of CEPs.

For instance, when applying the CEP to an R&D project in a research laboratory (the pattern’s context) in which management procedures are more informal, the management component of the

CEP is much too onerous for research teams (pattern's generic problem). Forces present here are the need for a minimal set of management procedures (to keep the project on track) and at the same time, the desire for leanness and freedom of action of the project's participants. The generic solution is to tailor the process by streamlining management tasks to remain within the process workflow (solution component of the pattern).

The concept of process patterns was introduced in the previous paragraph. While the primary goal of this document is to establish the CEP documentation framework, it was necessary to introduce process patterns as they are at the core of the PACEM project and now part of the toolset available to capability engineering method advisors to setup and document a capability engineering process.

In this section, the foundations of the CEP documentation framework have been laid. The concepts of agility, patterns and process patterns have been explained, as well as the theoretical abstraction-levels model, and some of the higher level components have also been presented. The framework content and its use will be the subject of the following sections.



## 3 Framework structure

---

### 3.1 General principles

Aladin generates the Web documentation from a method library. Method libraries contain method plug-ins and method configurations as illustrated in Figure 4. A method plug-in is a container for method packages that contains both the method and process content. A method content consists of content packages, standard categories and custom categories. Content packages include four types of elements: roles, tasks, work products and guidance. Categories serve to group related elements. There are five standard categories: disciplines, domains, work product types, roles sets and tools. There are three types of process elements: capability patterns, process patterns and delivery processes. A method configuration is the manifest of method plug-ins used to generate a specific instance of process guidance. It is built from library plug-in element subsets [25].

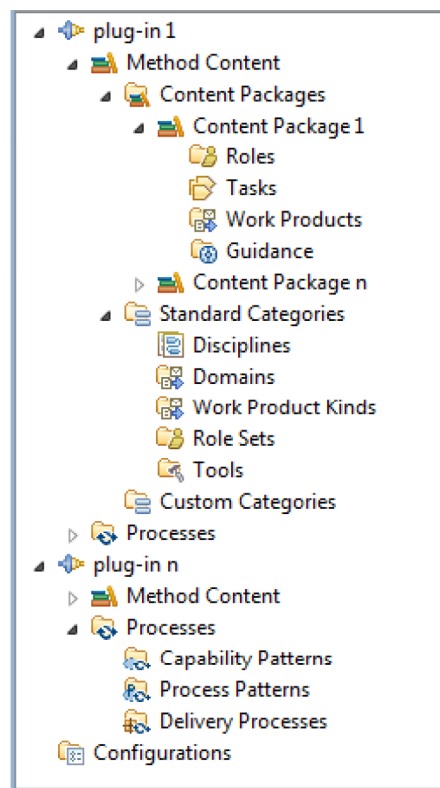


Figure 4: Method Library Structure

A method library can be structured in different ways. It could be composed of one or many plug-ins with their content organized in one or many content packages. Content packages may include very specific or very large sets of elements. Different approaches were examined prior to defining the CEP framework structure: the Unified Method Framework (UMF) and the Method Authoring Method (MAM) [22]; the one used in OpenUP (Open Unified Process) [23], a lean unified process and the SCRUM library [24]. UMF and MAM propose a modular approach for the library structure. This approach allows the reuse of common elements and true plug and play between

content from different authors. This leads to a library containing many specific plug-ins organized into four groups: core, practice, process and publish. Each group has separated areas of concern and specific ways to interact with other types. For the CEP Documentation Framework, it was found that although the UMF structure offers advantages, the overhead created may not be worth it in some contexts. A simple library structure is appropriate for processes with few tasks and roles. The needs are really the driver for the structure definition. Figure 5 depicts three examples of library structures. The right side structure describes all EPF practices that conform to UMF/MAM. Though the figure below does not show it, this library contains more than 40 plug-ins. The two other processes in Figure 5 have a simpler structure. SCRUM uses one plug-in while OpenUP is organized into three plug-ins. All structures are good if they meet the required needs.

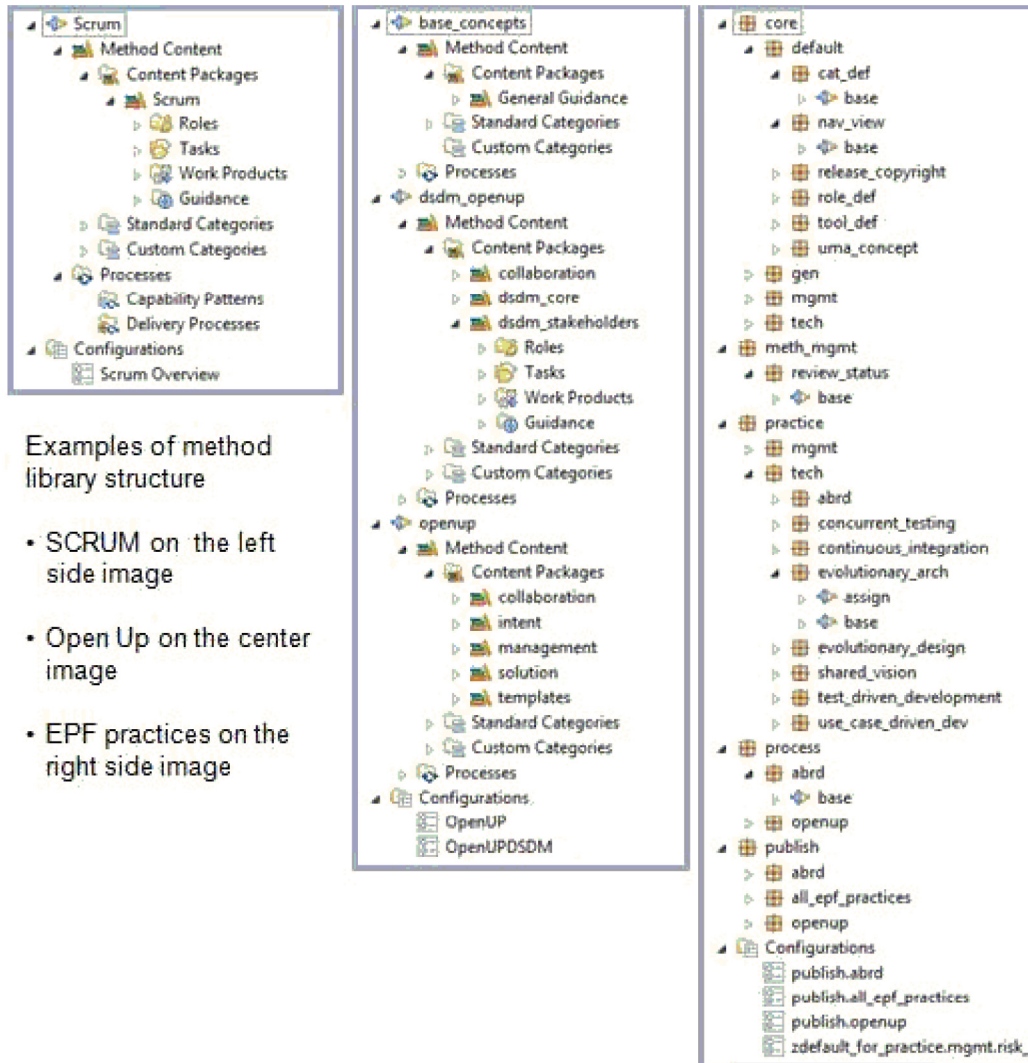


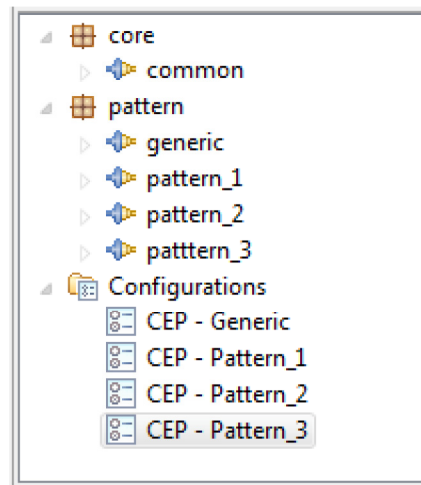
Figure 5: Examples of a Method Library Structure

## 3.2 The CEP library structure

Tests of different library structures from one plug-in to UMF-like structure reveal that a structure inspired by the reuse and modularity of UMF less its plug-and-play concept best met CEP needs, which were identified as:

- Easy reuse of generic CEP elements in different CEP flavour;
- Modularity;
- No circular references between plug-ins;
- Intuitive browsing;
- Logical division of packages and plug-ins;
- Easy retrieval of any element;
- Easy implementation of patterns; and
- Easy publishing of any CEP.

Figure 6 illustrates the resulting library structure. This structure contains two groups of plug-ins: core and pattern. The core group consists of one plug-in named common (referred as core.common) that includes all generic CEP method elements. No process elements are defined in this plug-in. The pattern group contains as many plug-ins as CEP adaptations. Each contains the definition of the method elements that differ from the generic ones as well as the description of the workflow. Finally, the configurations component of the structure includes a configuration for each CEP adaptation ready to be published.



*Figure 6: CEP High-Level Method Library Structure*

### 3.2.1 Core.common plug-in

The core.common plug-in serves as container for all generic CEP method elements and for Web documentation menu items. It does not include any process elements. CEP structure exploits the content packages, standard categories and custom categories of this plug-in.



### 3.2.1.1 Content packages

The method elements are organized into content packages. This facilitates the search of elements when modifications must be applied. Currently, the core.common plug-in includes 13 content packages. Table 1 describes each package with the type of element included (roles, tasks, work products or guidance). The first column presents the package system name, the second describes its content and the last column indicates the element types included.

*Table 1: Description of Content Packages*

Content Package	Description	Roles	Tasks	Work Products	Guidance
ACG_description	It describes the generic tasks, work products and guidance for the Analyze Capability Gap discipline.		✓	✓	✓
AFDO_description	It describes the generic tasks, work products and guidance for the Assess Force Development Options discipline.		✓	✓	✓
CEP_Acronyms_description	It lists acronyms applicable to CEP.				✓
CEP_Glossary_description	It lists definitions applicable to a generic CEP.				✓
CEP_Overview_description	It presents CEP foundations, principles and perspectives.				✓
CEP_Roles_description	It describes the basic role involved in a CEP initiative.	✓			
CEP_Stage_description	It contains a description of each CEP stage.				✓
Common_Work_Products_description	It includes work products that are used by more than one discipline.			✓	
Default_Publication_Package	It contains generic pages (e.g., About, Home, References, Example and CEP team pages) that any CEP Web documentation should publish. Copyright information is also found in this package.				✓
DOA_description	It describes the generic tasks, work products and guidance for the Develop Operational Architecture discipline.		✓	✓	✓
DSOSA_description	It describes the generic tasks, work products and guidance for the Develop SoS Architecture discipline.		✓	✓	✓
MEE_description	It describes the generic tasks work products and guidance for the Manage Engineering Effort discipline.		✓	✓	✓

### 3.2.1.2 Standard categories

The CEP structure uses standard categories to group elements and to facilitate their presentation in Web documentation. The display of a category entails that all of its elements are displayed. Table 2 shows the standard categories utilized in core.common and their content. Disciplines, domains and role sets are used to categorize tasks, work products and roles respectively. The first column indicates the type of category, the second the name as it appears in Aladin Navigator and the third describes its content.

*Table 2: Standard Categories Organization*

<b>Standard Categories</b>	<b>Name</b>	<b>Category Content</b>
Disciplines	Analyze_Capability_Gap_discipline	Description of the Analyze Capability Gap discipline and a list of its tasks.
	Assess_Force_Development_Options_discipline	Description of the Assess Force Development Options discipline and a list of its tasks.
	Develop_Operational_Architecture_discipline	Description of the Develop Operational Architecture discipline and a list of its tasks.
	Develop_SoS_Architecture_discipline	Description of the Develop SoS Architecture discipline and a list of its tasks.
	Manage_Engineering_Effort_discipline	Description of the Manage Engineering Effort discipline and a list of its tasks.
Domains	Force_Development_Options_domain	List of artifacts related to the Assess Force Development Options discipline.
	Integrated_Dictionary_domain	List of common artifacts to each discipline: glossary of terms, list of symbols, etc.
	Management_domain	List of work artifacts related to the Manage Engineering Effort discipline.
	Operational_Architecture_domain	List of artifacts related to the Develop Operational Architecture discipline.
	Requirements_domain	List of artifacts related to the Analyze Capability Gap discipline.
	SoS_Architecture_domain	List of artifacts related to the SoS Architecture for the recommended/ chosen options.
Role Sets	CEP_Designers_Role_Sets	List of roles that draw Force Development Options for the capability.
	CEP_Stakeholders_Role_Sets	List of roles having a claim in the capability for which options are developed.

### 3.2.1.3 Custom categories

Custom categories are used in the CEP to define the navigation structure of the Web documentation. A custom category can include any element types: method elements, process elements, standard categories and custom categories. Figure 7 illustrates CEP Custom Categories and their constituting elements. Blue boxes represent Custom Categories, white ones represent Standard Categories. Blue text describes the leaves of the menu tree. The text can indicate a specific method element, how a category is filled by Aladin or how a category is used. Currently, CEP uses a root category named `Navigation_view_generic` which contains six custom categories and two guidance elements to build its main menu. The composition of each category of this structure represents a menu item.



Figure 7: CEP Custom Categories

## 3.2.2 Pattern plug-ins

Pattern plug-ins play two main roles: 1) define method elements that differ from generic to adapted CEP; and 2) define the process pattern. Each plug-in contains the definition of the method elements that differ from the generic ones as well as the description of the workflow. This implies that a pattern plug-in may comprise method and process content as opposed to the `core.common` plug-in which contains only method content.

### 3.2.2.1 Method content

Method content of a pattern plug-in that documents differences from `core.common` in terms of content packages, standard categories and custom categories. For each of them, their structure

defined in 3.2.1 is reproduced only for the part affected by changes. In content packages, changes consist of:

- addition of a task, role, work product or guidance; or
- modification to an existing method element (task, role, work product or guidance).

Standard categories in a pattern plug-in document changes in a category content. They are not affected by a modification to an existing task, artifact or role. The reproduction of a discipline, domain or role set occurs in the following context:

- addition or removal of a task in a discipline;
- addition or removal of an artifact in domain; or
- addition or removal of a role in a role set;

Custom Categories in a pattern plug-in documents changes in the navigation structure of the Web documentation. They are not affected by a modification to existing method elements. The reproduction of a custom category occurs in the following context:

- modification of its description (e.g., presentation name); or
- modification of its content (addition, deletion or ordering of elements).

In summary, the size of method content of a pattern plug-in depends on the importance of changes from core.common.

### 3.2.2.2 Process content

Process content is used 1) to describe the specificity of a CEP adaption related to its generic version and 2) to document the workflow of activity for CEP life cycle. Process Content consists of three items: capability patterns, process patterns and delivery processes. The following paragraphs provide an overview of the role of capability patterns and process patterns. This document does not address delivery processes as they are not used in this framework.

Two types of elements exist in capability patterns: process packages and capability patterns. Process packages contain capability patterns and serve to organize them. This framework proposes six packages to structure the capability patterns. Their organization follows CEP disciplines and life cycle as shown in Table 3. Capability pattern describes activities, iterations and stages with their corresponding workflow. Generic CEP contains a complete set of capability patterns to describe the CEP life cycle while CEP adaptation contains a subset describing what differs from generic CEP.

*Table 3: Process Packages Description*

Process Packages	Description
ACG	It includes the activities related to the Analyze Capability Gap discipline.
AFDO	It includes the activities related to the Assess Force Development Options discipline.

DOA	It includes the activities related to the Develop Operational Architecture discipline.
DSoSA	It includes the activities related to the Develop SoS Architecture discipline.
MEE	It includes the activities related to the Manage Engineering Effort discipline.
stages-iterations	It defines the content of each CEP iteration and stage

The Process Patterns folder contains the description and workflow of a CEP adaptation. This element describes the specificity of the CEP adaptation in relation to a CEP generic version. It is used by a search engine to find a CEP adaptation matching definite criteria.

### 3.2.2.3 Library configurations

The last type of component found in the CEP library structure is called configurations. Configurations contain publishing information about the plug-in that shall be included and the navigation views. This framework recommends creating a configuration for each CEP adaptation. Each configuration comprises at least one core.common plug-in and one pattern plug-in.

In summary, CEP library structure is organized in two groups of plug-ins called core and pattern. The core group comprises a unique plug-in where all generic method elements are organized as specified by the framework. The pattern group includes one plug-in by CEP adaptation. Each plug-in defines the specifics of the method element for its CEP adaptation as well as the activities and workflows. These plug-ins also follow structural rules.



## 4 Framework usage

---

The framework focuses on how to document and publish Web documentation for a complete capability engineering process using Aladin. The intent is to explain how to complete the information, not how to use the Aladin software. The EPF composer manual [25] can serve as a guide to using Aladin. The production of the Web documentation requires the use of some sequences, guidance, rules and checklists. This section starts with the completion of a generic CEP even though all its elements are provided with the framework. The comprehension of how a generic CEP was introduced in the framework is important for the second part of this section: CEP adaptation.

### 4.1 Creation of a generic CEP

The production of a generic CEP requires populating the structure built within Aladin with respect to the CEP framework completion sequence, element description and validation rules. The framework helps to enforce coherence and quality of the Web documentation. The section starts with an overview of the steps necessary to produce a site. Then, general rules are presented, followed by specific guidance to document method elements, activity and workflows. Finally, the section ends with some validation checklists and publishing guidance.

#### 4.1.1 Completion sequence

The sequence to produce the Web documentation for a generic CEP involves four major steps:

- building the framework structure;
- entering each generic method element in the core.common plug-in;
- completing the appropriate parts of the pattern.generic plug-in; and
- publishing the site.

The sequence presented below assumes the framework structure is not in place even though generic CEP is ready to publish in the framework. The purpose of this assumption is to present the sequence from start to end.

##### 1. Build the basic structure

- a. Create the plug-in for method content. Add a new plug-in, name it core.common and indicate a brief description (optional) and a version.
- b. Create a plug-in for process content. Add a new plug-in, name it pattern.generic and indicate a brief description and a version (optional).
- c. Create a method configuration. Add a new method configuration, name it CEP Generic and insert a brief description.

## 2. Populate core.common

- a. Define content packages: Add each content package indicated in Table 1 and name them accordingly. Give them a presentation name and insert a brief description.
- b. Define standard categories: Add five disciplines, five domains and two role sets. Name the categories as indicated in Table 2. Insert a presentation name that corresponds to their name minus the suffix and the underscore characters (e.g., Analyze Capability Gap), a brief description and a main description (optional).
- c. Populate the content packages. Add the relevant method elements in each content package in this order:
  - i. Guidance
  - ii. Artifacts
  - iii. Deliverables
  - iv. Roles
  - v. Tasks

Details on how to complete each type of method element are provided in 4.1.3.

- d. Indicate copyright. Insert in the description of the core.common plug-in the guidance defined as copyright in the publication package. This copyright acts as the default copyright for each element of core.common if nothing else is indicated.
- e. Define custom categories. Create each category illustrated in Figure 7 starting with the root category and the name, and build them as indicated in the figure.

## 3. Populate pattern.generic

- a. Define process packages. Under capability patterns, add each process package indicated in Table 3, and name them accordingly.
- b. Define activities. For each process package, with the exception of stages-iteration, build the activities relevant to the package using the capability pattern. Provide a relevant name for the capability pattern and associate it with CEP – Generic Configuration. Details on how to complete activity elements are provided in 4.1.4.
- c. Define iterations-stages. Under the stages-iteration package, build each iteration and stage of CEP life cycle using a capability pattern. Provide a relevant name for the capability pattern and associate it with CEP – Generic Configuration. Insert the activities relevant in each stage and iteration, and generate the

workflows. Details on how to complete stage and iteration elements are provided in 4.1.5.

#### 4. Publish site

- a. Launch Publish function. Complete navigation menu. Confirm configuration. Select publishing options. Generate the Web documentation. Details on how to publish CEP life cycle documentation are provided in 4.1.6.

### 4.1.2 General rules

The framework recommends applying some rules while populating its structure. These rules serve to enforce coherence, homogeneity and quality of the content as well as conformance with the CEP foundation. This section presents rules applying to the overall framework. Many rules are inspired from the Method Authoring Method (MAM) [22].

#### Structural rules

- Changes to the framework must respect the CEP foundation.
- Preserve CEP basic definitions, acronyms and overview material.
- Avoid hyperlinks between content packages.
- Avoid circular reference between plug-ins.
- Avoid modifying core.common and pattern.generic plug-ins as they serve as references for CEP adaptation. Apply modifications only if changes are required to the generic CEP.
- Avoid the creation of new content packages, standard categories and custom categories. Use the packages and categories already defined.
- Keep process content of core.common empty.

#### Writing practices

- Make sure to put a period at the end of each sentence.
- Check for general spelling and spacing errors.
- Apply the five Cs of good writing: clear, concise, comprehensible, consistent and correct.
- Use active form and present tense.
- Ensure coherent use of capital letters.

#### Formatting practices for fields with rich text editor (RTE) capacity

- Use GIF and JPG formats for images.
- Ensure coherent use of bold, underline and italic.
- Ensure coherent use of font size and type.
- Ensure formatting homogeneity.



- Select type method element when adding a link to an element.
- Only use formatting features offered by the RTE.

Rules for a common field to each type of element (from MAM).

- Name: Choose a significant name with no spaces. Spaces can be replaced by underscores. A suffix indicating the type of element should be added to the name, e.g., Develop\_Operational\_Concept\_task.
- Presentation name: Choose a significant name with appropriate spelling. Avoid acronyms. The element presentation name and name should be consistent.
- Brief description: Explain in few sentences what the element is without stating the obvious or repeating what can be deduced from the element's name. Do not include the name of the element in the description; instead refer to the element by its type (this role, this task, this guidance, etc.). Avoid repeating the brief description in other fields of the method element.
- Guidance: Include in this field all relevant guidance for the element. A reference for each guidance specified should appear in one of the RTE fields of the element (e.g., main description, steps).

#### **4.1.3 Method elements completion rules**

The description of each method element is entered through specific editor (e.g., role editor, task editor). These editors are composed of tabs which are divided into sections as shown in Figure 8. Two tabs are common in their usage to any type of method element: Guidance and Preview (circled in orange in the figure below). The Guidance tab offers a mechanism to add and briefly described guidance relevant to the method element. A click on the Preview tab shows the element as it will look once published. It is useful to verify element information. As both tab completion is intuitive, this document does not provide further explanation.

Guidance (Concept): Requirements\_concept

**General Information**  
Provide general information about this concept.

Name: Requirements\_concept

Presentation name: Requirements

Type: Concept [Change Type...](#)

Brief description: In general term, the concept of requirement refers to "a documented representation of a condition or a capability needed by a user to solve a problem or achieve an objective". A distinction is often made with a raw requirement (often called need) that refers

**Detail Information**

**Version Information**  
Provide version information about this concept.

Version:

Change date: Thursday, October 28, 2010

Change description:

Authors:

Copyright: © CEP\_copyright [Select...](#) [Deselect](#)

**Content Variability**  
Specify how this concept relates to another concept.

Variability type: Not applicable

Base:

[Select...](#)

**Icon**  
Customize the icons for this concept.

Shape Icon Preview: [Select...](#) [Clear](#)

Node Icon Preview: [Select...](#) [Clear](#)

Description [Guidance](#) [Preview](#)

Figure 8: Editor for Method Content Element

To limit repetitions in the description of editors, the completion of sections that are identical for any method elements are described below. These sections are circled in red in Figure 8.

- **Version Information.** This section includes internal (not published) and published fields.
  - ♦ **Version:** This internal field is not mandatory.
  - ♦ **Change date:** This internal field is not mandatory. A click on the field enters today's date. It cannot be removed.
  - ♦ **Change description:** This internal field is not mandatory. Enter any free-form text to describe changes.
  - ♦ **Authors:** This internal field is not mandatory. Enter the author(s) of the content.
- **Copyright:** This mandatory field uses the copyright entered at the plug-in level when left empty. A guidance describing the copyright needs to be inserted into this field if the copyright information at the plug-in level is inexistent or irrelevant.

- **Content Variability:** This section concerns method element adaptation. Its completion is explained in 4.2.
- **Icon:** This section, available in most method elements, allows the user to change the default icons for the element. Two icons may be modified.
  - ♦ **Shape icon:** This icon is displayed in the upper left corner of the content page when published.
  - ♦ **Node icon:** This icon is displayed in the navigation view of the published site.

The next paragraphs present the specific editor for each method element. A table introduces tabs included in each editor with some circled fields. Version, content variability and icon sections from the Description tab are collapsed to reduce the screenshot size. Red-framed boxes indicate mandatory fields while orange-framed boxes indicate recommended fields. Fields with no framed boxes do not have to be completed in accordance with the CEP framework. For field completion instructions, follow the table. Field completion should comply with the general rules contained in 4.1.2.

#### 4.1.3.1 Guidance

Guidance is a type of method element that can be associated with any method element to complement its description or provide further information on its why and how. Each editor of any method type contains a tab to enter guidance. Even if Aladin offers other kinds of guidance, the framework restricts their use to the seven types presented in Table 4. Each column presents a snapshot of a tab for a specific guidance editor.

- **First column:** Description tab for guidance for supporting material and guideline and concept types.
- **Second column:** Description tab for guidance for example and template types.
- **Third column:** Description tab for guidance for term definition type.
- **Fourth column:** Description tab for guidance for checklist type.
- **Fifth column:** Check item tab for guidance for checklist type.

For the Description tab of any of these editor types (Supporting Material, Guideline, etc.), the fields should be completed as follows:

*Table 4: Description of Guidance Editor*

Description tab				Check items tab
Supporting Material, Guideline and Concept guidance	Example and Template guidance	Term Definition guidance	Checklist guidance	Checklist guidance

<p>▼ General Information Provide general information</p> <p>Name: <input type="text"/></p> <p>Presentation name: <input type="text"/></p> <p>Type: <input type="text"/></p> <p>Brief description: <input type="text"/></p> <p>▼ Detail Information Provide detailed information</p> <p>Main description: <input type="text"/></p>	<p>▼ General Information Provide general information</p> <p>Name: <input type="text"/></p> <p>Presentation name: <input type="text"/></p> <p>Type: <input type="text"/></p> <p>Brief description: <input type="text"/></p> <p>Attached file(s)/URL(s): <input type="text"/></p> <p>▼ Detail Information Provide detailed information</p> <p>Main description: <input type="text"/></p>	<p>▼ General Information Provide general information</p> <p>Name: <input type="text"/></p> <p>Presentation name: <input type="text"/></p> <p>Type: <input type="text"/></p> <p>Brief description: <input type="text"/></p> <p>▼ Detail Information Provide detailed information</p> <p>Main description: <input type="text"/></p>	<p>▼ General Information Provide general information</p> <p>Name: <input type="text"/></p> <p>Presentation name: <input type="text"/></p> <p>Type: <input type="text"/></p> <p>Brief description: <input type="text"/></p> <p>▼ Detail Information Provide detailed information</p> <p>Main description: <input type="text"/></p>	<p>▼ Check Items Specify the check items</p> <p>Check Items</p> <ul style="list-style-type: none"> <li>◆ Requirement</li> <li>◆ All internal</li> <li>◆ Requirement</li> <li>◆ Requirement</li> <li>◆ Requirement</li> </ul> <p>Name: <input type="text"/></p> <p>Requirements are</p> <p>Description: <input type="text"/></p> <p>Interoperability r</p>
---	--	---	---	---

#### Mandatory fields:

- Name: In this internal field, provide a significant name for the guidance. Add a suffix to the name indicating the nature of the element (e.g., .checklist or .guidance).
- Presentation name: In this published field, provide the guidance name as it should appear in the Web documentation.
- Brief description: In this published field, provide one or two sentences describing what the guidance is.
- Main description (only applicable to a term definition guidance type): In this published field, provide a well-structured definition of the term or acronyms and its meaning.

#### Recommended fields:

- Main description (applicable to a guidance type listed in column 1, 2 or 4): In this published field, provide an extended description of the guidance beyond what is covered in other fields. Text in the field can be formatted and complemented with images and links.
- Attached file(s)/URL(s) (applicable to Example and Template guidance types): In this field, insert file or link to information not defined inside the framework.

For the Check Items tab (Column 5) of the Checklist editor, the fields should be completed as indicated below and comply with the rule numbers in parentheses.

#### Mandatory field:

- Name: In this published field, provide a short description for the item. Once entered, the item appears in the Check Items box.


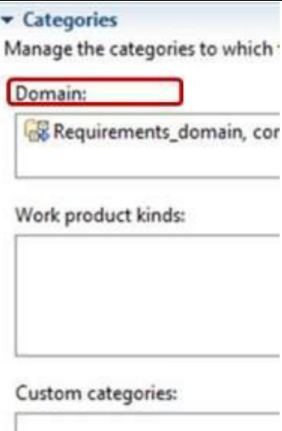
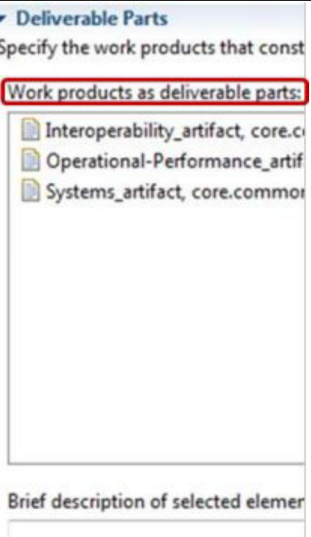
Recommended field:

- Description: In this published field, provide an extended description of the item. Text in the field can be formatted and complemented with images and links.

#### 4.1.3.2 Work products

Aladin proposes three types of work products: Artifact, Deliverable and Outcomes. This document elaborates on artifact and deliverables as they are used in CEP Documentation Framework. The editor has four (Artifact) or five (Deliverable) tabs. Table 5 illustrates these tabs excluding the Guidance and Preview tabs for this editor. Artifact should be completed before deliverable to limit going back and forth between descriptions.

Table 5: Description of Work Product Editor

Description tab	Categories tab	Deliverable parts tab
Artifact and deliverable work products		Deliverable
		

Mandatory fields:

- Name: In this internal field, provide a significant name for the artifact or deliverable.
- Presentation name: In this published field, provide the work product name as it should appear in the Web documentation.
- Brief description: In this published field, provide one or two sentences describing what artifact or deliverable it is.
- Work product as deliverable parts: In this published field, include every specific artifact that is part of the deliverable. Do not include as deliverable parts the artifacts from the Integrated dictionary domain: Glossary\_of\_terms and List\_of\_symbols.



- Domain: In this field, assign the relevant category for the artifact. Domains are used in the framework exclusively for artifacts. Do not include more than one domain per artifact. Do not categorize deliverable into domain.

### 4.1.3.3 Roles

*The role editor consists of three tabs, excluding the Guidance and Preview tabs: Description, Work Products, and Categories.*

Table 6 shows a sample of each tab without the Version Information, Content Variability and Icon sections of the Description tab.

*Table 6: Description of Role Editor*

Description tab	Work Products tab	Categories tab
<div> <div>General Information</div> <div>Provide general informat</div> <div>Name:</div> <div>Presentation name:</div> <div>Brief description:</div> </div> <div> <div>Detail Information</div> <div>Provide detailed informa</div> <div>Main description:</div> <div>Key considerations:</div> </div> <div> <div>Staffing Information</div> <div>Provide staffing informa</div> <div>Skills:</div> <div>Assignment approaches:</div> <div>Synonyms:</div> </div>	<div> <div>Work Products:</div> <div>Specify work produ</div> <div>Responsible for:</div> </div> <div> <div>Work products th</div> <div>Architecture_</div> <div>Behaviours_M</div> <div>Functions_Al</div> <div>Functions_De</div> <div>Glossary_of_t</div> <div>Implementat</div> <div>Interoperabili</div> </div> <div> <div>Brief description c</div> </div>	<div> <div>Categories:</div> <div>Manage the categorie</div> <div>Role sets:</div> <div>CEP_Designers_</div> </div> <div> <div>Custom categories:</div> </div> <div> <div>Brief description of :</div> </div>

Mandatory fields:

- Name: In this internal field, provide a significant name for the role.
- Presentation name: In this published field, provide the role name as it should appear in the Web documentation.

- **Brief description:** In this published field, provide one or two sentences describing what the role is.
- **Responsible for:** Include in this field all work products (artifacts and deliverables) under the role responsibility. The responsibility for work products cannot be shared among roles. A role may contribute to a work product without having its responsibility.
- **Role sets:** In this field, assign the relevant category (CEP Designers or CEP Stakeholders) to this role. Do not include more than one role set for a role.

Recommended fields:

- **Main description:** In this published field, provide an extended description of the role beyond what is covered in other fields. Text in the field can be formatted and complemented with images and links.
- **Skills:** In this field, provide the essential and desirable knowledge and experience as well as competencies required by the role.
- **Assignment approaches:** In this field, provide the involvement (e.g., part time, full time) required by the role at each CEP stage and iteration.

Automatically completed field:

- **Work products** that are output of tasks performed by this role. Aladin extracts the information displayed in this field from the task description.

#### **4.1.3.4 Task**

Table 7 presents screen snapshots of tabs (excluding the Guidance and Preview tabs) that make up the task editor. Tasks are the most complex element to be described in the method content. They should be the last element type completed as they use other elements in their definition.

Table 7: Description of Task Editor

Description tab	Steps tab	Roles tab	Work Products tab	Categories tab
<p>▼ General Information Provide general information</p> <p>Name: <input type="text"/></p> <p>Presentation name: <input type="text"/></p> <p>Brief description: <input type="text"/></p> <p>▼ Detail Information Provide detailed information</p> <p>Purpose: <input type="text"/></p> <p>Main description: <input type="text"/></p> <p>Key considerations: <input type="text"/></p> <p>Alternatives: <input type="text"/></p>	<p>▼ Steps Specify the steps to perform the task</p> <p>Steps:</p> <ul style="list-style-type: none"> <li>Identify the strategy</li> <li>Analyse the strategy</li> <li>Derive the inputs</li> </ul> <p>Name: <input type="text"/></p> <p>Identify the strategy</p> <p>Description: <input type="text"/></p> <p>Establish a list of steps but additional types</p> <ul style="list-style-type: none"> <li>Political factors</li> <li>Economic factors</li> </ul>	<p>▼ Roles Assign the roles to perform the task</p> <p>Primary performers:</p> <ul style="list-style-type: none"> <li>Operational_Architect</li> <li>Requirements_Analyst</li> </ul> <p>Additional performers:</p> <ul style="list-style-type: none"> <li>External_Users_role</li> <li>Operational_Representative</li> <li>PRICIE_Representative</li> </ul> <p>Brief description of selected roles</p>	<p>▼ Work Products Specify the input and output products</p> <p>Mandatory inputs:</p> <ul style="list-style-type: none"> <li>Mandate_artifact</li> <li>Stakeholders_analysis</li> </ul> <p>Optional inputs:</p> <p>Outputs:</p> <ul style="list-style-type: none"> <li>Strategic_Factors</li> </ul> <p>Brief description of work products</p>	<p>▼ Categories Manage the categories</p> <p>Disciplines:</p> <ul style="list-style-type: none"> <li>Analyse</li> </ul> <p>Custom categories</p> <p>Brief description</p>

Mandatory fields:

- Name (Description tab): In this internal field, provide a significant name for the task that is composed of a verb and a complement. Choose the verb carefully; it should indicate the “what” of the task. See Annex B for a list of verbs.
- Presentation name: In this published field, provide the task name as it should appear in the Web documentation. The name is composed of a verb carefully chosen and a complement; it should indicate the “what” of the task. See Annex B for a list of verbs.
- Brief description: In this published field, provide one or two sentences describing what the task is. The description should not include any directives on how to perform the task.
- Purpose: In this published field, provide a short sentence starting with “to” that indicates the goal of the task, e.g. “To determine a validated set of operational and performance parameters criteria.”
- Name (Steps tab): In this published field, enter a significant name for each step necessary to perform the task. The step name should consist of a verb and a complement. The steps must be organized in the order of their execution.
- Primary performers: In this published field, enter roles that contribute to the task.



- Additional performers: In this published field, enter roles that can be consulted for task execution.
- Mandatory inputs: In this published field, include each work product necessary to perform the task. Usually, work products will consist of artifacts.
- Outputs: In this published field, include the work products resulting from the task. Usually, work products will consist of artifacts.
- Disciplines: Assign in this field the relevant category (ACG, AFDO, DOA, DSoSA, MEE) to this task. Do not include more than one discipline in a task.

Recommended fields:

- Description: In this published field, provide a description for the step.
- Optional inputs: In this published field, include each work product that the task may require. Usually, work products will consist of artifacts.

#### 4.1.4 Activity completion rules

Activities are documented via the Capability Pattern editor. This editor is composed of five tabs which are divided into sections. Two tabs require information in this document framework: Description (Figure 9) and Work Breakdown Structure (Figure 10). The grouping of tasks into activity is the main focus of activity description.

Few fields need to be completed in the Description tab. The name and configuration will already appear as the information is requested at the creation of a capability pattern. The editor allows the user to revise them and enter further information.

The screenshot shows the 'Capability Pattern: Define\_Decision\_Criteria\_activity' editor. It has two main sections: 'General Information' and 'Detail Information'.

**General Information:** This section contains fields for 'Name' (filled with 'Define\_Decision\_Criteria\_activity'), 'Presentation name' (filled with 'Define Decision Criteria'), 'Brief description' (empty), and 'Purpose' (empty). The 'Name' and 'Presentation name' fields are highlighted with red boxes.

**Detail Information:** This section contains a 'Configuration' subsection. It has a 'Configurations' list with 'CEP 2011 (default)' selected. To the right of the list are buttons for 'Add...', 'Remove', and 'Make Default'. Below the list is a 'Description' field filled with 'Generic CEP 2011'. The 'Configurations' label is highlighted with a red box.

At the bottom, there is a tabbed interface with five tabs: 'Description' (active), 'Work Breakdown Structure', 'Team Allocation', 'Work Product Usage', and 'Consolidated View'.

*Figure 9: Description Tab for Activity Edition*

The framework recommends completing the fields listed below. The general rules presented in section 4.1.2 as well as those indicated in the field description should be respected.

- **Name:** In this internal field, provide a significant name for the activity that is composed of a verb and a complement. Choose the verb carefully; it should reflect the activity goal. See Annex B for a list of verbs.
- **Presentation name.** In this published field, provide the activity name as it should appear in the Web documentation. The name is composed of a carefully chosen verb and a complement and should reflect the activity goal. See Annex B for a list of verbs.
- **Configuration:** In this internal field, include a valid configuration for the activity. This field should not contain more than one configuration according to the documentation framework.

The activity definition continues with the inclusion of tasks in the Work Breakdown Structure tab of the Capability Pattern editor. Each task composing the activity is added into the editor. A task belongs to only one activity. Activities do not cross disciplines. They may include tasks performed at different stages but each task must belong to the same discipline.

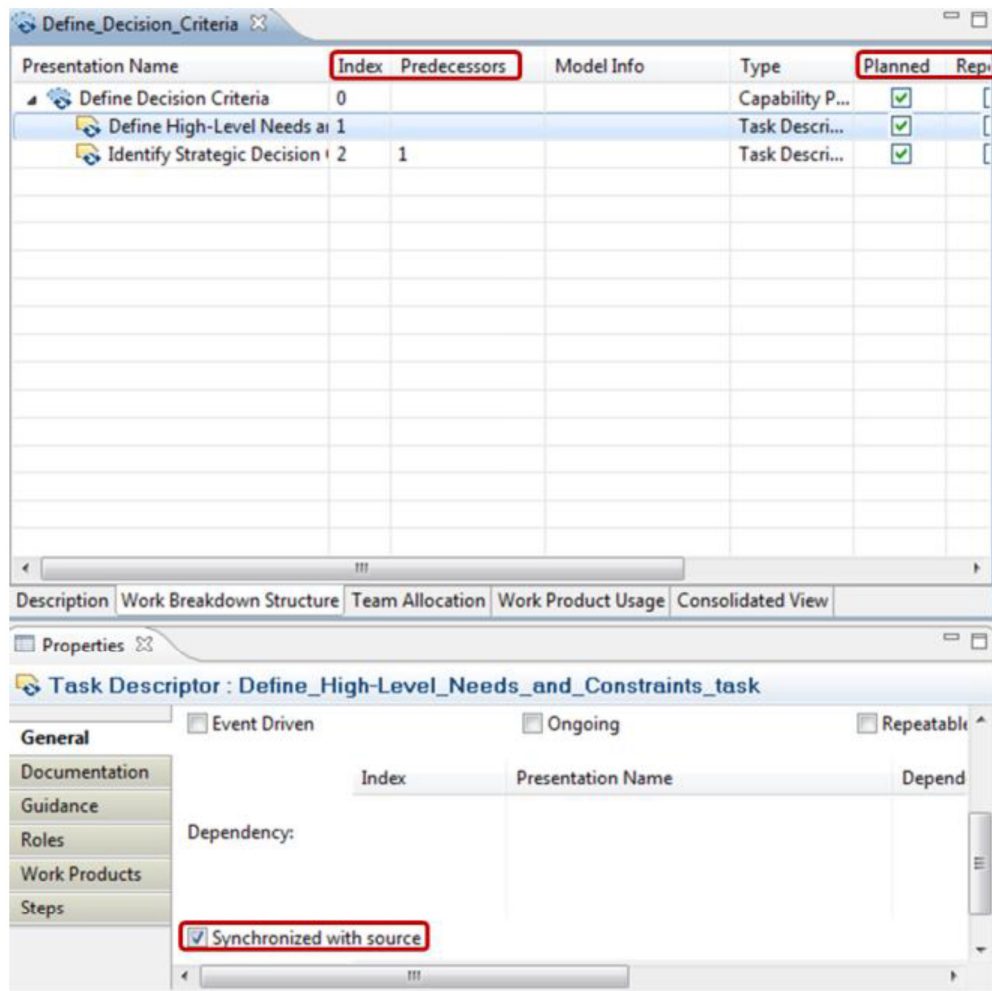


Figure 10: Work Breakdown Structure Tab for Activity Editing

The work breakdown structure is built following this sequence:

1. Add task from the Method Content: Choose tasks that are from the same discipline as the activity. This is done from the contextual menu on the activity line.
2. Check the appropriate boxes for the task: The framework uses Planned, Repeat or Ongoing.
3. Revise the index and predecessors of each task: This represents the sequence of tasks within the activity.
4. Check the Synchronized with source box as a property for each task: This synchronizes the task included in the activity with the task in the Method Content.
5. Open the Activity Detail Diagram to generate the diagram and adjust it if required: This generation is done from the contextual menu on the activity line.

6. Synchronize the activity with the Method Content to ensure the latest updates of the method elements are included: This synchronization starts by selecting Default Synchronization from the Method Content on the activity line in the contextual menu.

#### 4.1.5 Stage/iteration rules

The framework uses capability patterns to define CEP iterations and stages. Figure 11 shows the capability patterns representing stages and iterations needed to document the CEP delivery process. The terms within red boxes designate stages, the others designate iterations. The text in parentheses specifies how the stage or iteration is constructed. The construction can be directly from assembling and sequencing activities (A) or from iterations. The comprehension stage is the combination of Comprehension 1 (C1) and Comprehension 2 (C2) iterations. The same applies for the elaboration stage which is built from three iterations (E1 + En + Efinal). Both constructions are documented via the capability pattern editor. Each capability pattern has a name and presentation name, and is associated with a configuration. This information is completed through the Description tab of the editor (see Figure 9).

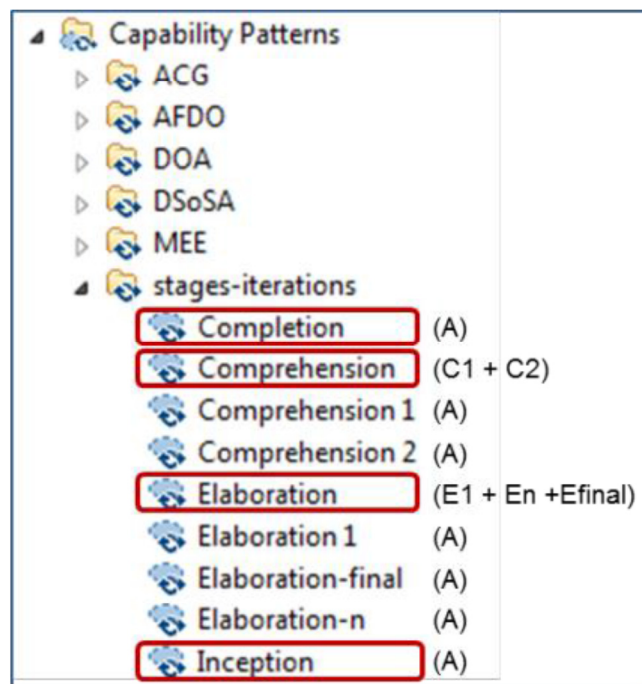


Figure 11: CEP Iterations and Stages

The Work Breakdown Structure tab is used to detail the structure of iterations and stages. The procedure to define iterations and stages marked with an (A) differs from that for the Comprehension and Elaboration stages. Thus, the instructions presented are twofold. First, the procedure to build the item marked with an A in Figure 11 is expounded. Then, the explanation to construct the comprehension and elaboration stages are specified.

Figure 12, which represents the first iteration of comprehension, is an example of how iterations are built.

The screenshot displays a software interface with a hierarchical tree of activities on the left and a detailed properties editor on the right.

**Activity Tree:**

Presentation Name	Index	Predecessors	Model Info	Type	Planned	Repeat...	Multipl...	Ongoing
Comprehension 1	0			Capability P...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Define Evaluation Criteria	1		extends 'Define_Eva...	Activity	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Identify Risks	2			Task Descri...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Develop CEDF Performa	3	2		Task Descri...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Refine CEDF Performa	4			Task Descri...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Define Operational Option	5	11	extends 'Define_Op...	Activity	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Outline Operational O	6			Task Descri...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Describe CEDF Model	7	15,5,1,19,31	extends 'Describe_...	Activity	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Develop CEDF Model	8			Task Descri...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Refine CEDF Model	9			Task Descri...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Optimize FDO	10	9		Task Descri...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Describe Operational Con	11		extends 'Describe_...	Activity	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Develop Overarching C	12			Task Descri...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Analyse Operational S	13			Task Descri...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Refine Overarching Op	14			Task Descri...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Describe Operational Requ	15	11	extends 'Describe_...	Activity	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Refine Interoperability	16			Task Descri...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Refine Operational Re	17			Task Descri...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Refine Performance Re	18			Task Descri...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Describe Relevant Systems	19		extends 'Describe_R...	Activity	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Identify Existing System	20			Task Descri...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Analyse Existing System	21	20		Task Descri...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Analyse Potential System	22	23		Task Descri...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Identify Potential System	23			Task Descri...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Manage Iteration	24		extends 'Manage_It...	Activity	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Assess and Control Effi	25			Task Descri...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Organize Next Iteration	26	7,24,29	extends 'Organize_...	Activity	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Plan Next Iteration	27			Task Descri...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Close-Out	28	27		Task Descri...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

**Properties Editor: Activity: Define\_Evaluation\_Criteria**

**General**

☐ Optional ☐ Multiple Occurrences ☒ Planned ☐ Suppressed

☐ Event Driven ☐ Ongoing ☐ Repeatable

**Dependency:**

Index	Presentation Name	Dependency

**Model information:** extends 'Define\_Evaluation\_Criteria, pattern.generic'

**Activity Type:** Activity Change Type...

Figure 12: Description of an Iteration

The sequence to assemble this type of item consists of:

1. Adding all relevant capability patterns into the iteration: This is done by selecting Apply Pattern/Extend from the contextual menu on the iteration/stage line. The selected pattern represents one of the activities already defined. Only an activity (capability pattern) should be added to an iteration or a stage. Tasks should never be added directly.
2. Changing each capability pattern to an activity in the Properties editor.



3. Checking the appropriate boxes for each type of activity: The framework uses Planned, Repeat or Ongoing.
4. Revising the index and predecessors of each activity: This represents the sequence of activities within the iteration.
5. Suppressing tasks that do not need to be performed within the iteration: The activity may include tasks that are not relevant for a specific stage or iteration. If this is the case, suppress the task using the contextual menu on the task line. Suppressed tasks appear in grey in the work breakdown structure.
6. Generating the workflow diagram and adjust it if required: This generation is done by selecting Open Activity Diagram from the contextual menu on the iteration line. Once the diagram is open, it can be improved by:
  - Adding the Start and End Node symbol;
  - Adding the Fork Node and Join Node;
  - Aligning activities; or
  - Placing emphasis on a parallel (simultaneous) work stream. Vertical placement of activities can be adjusted to reflect the approximate sequence of work. Tasks that can start later are placed below tasks starting earlier.

Figure 13 illustrates the result of applying these actions. Part A presents the generated workflow while part B presents the improved workflow.



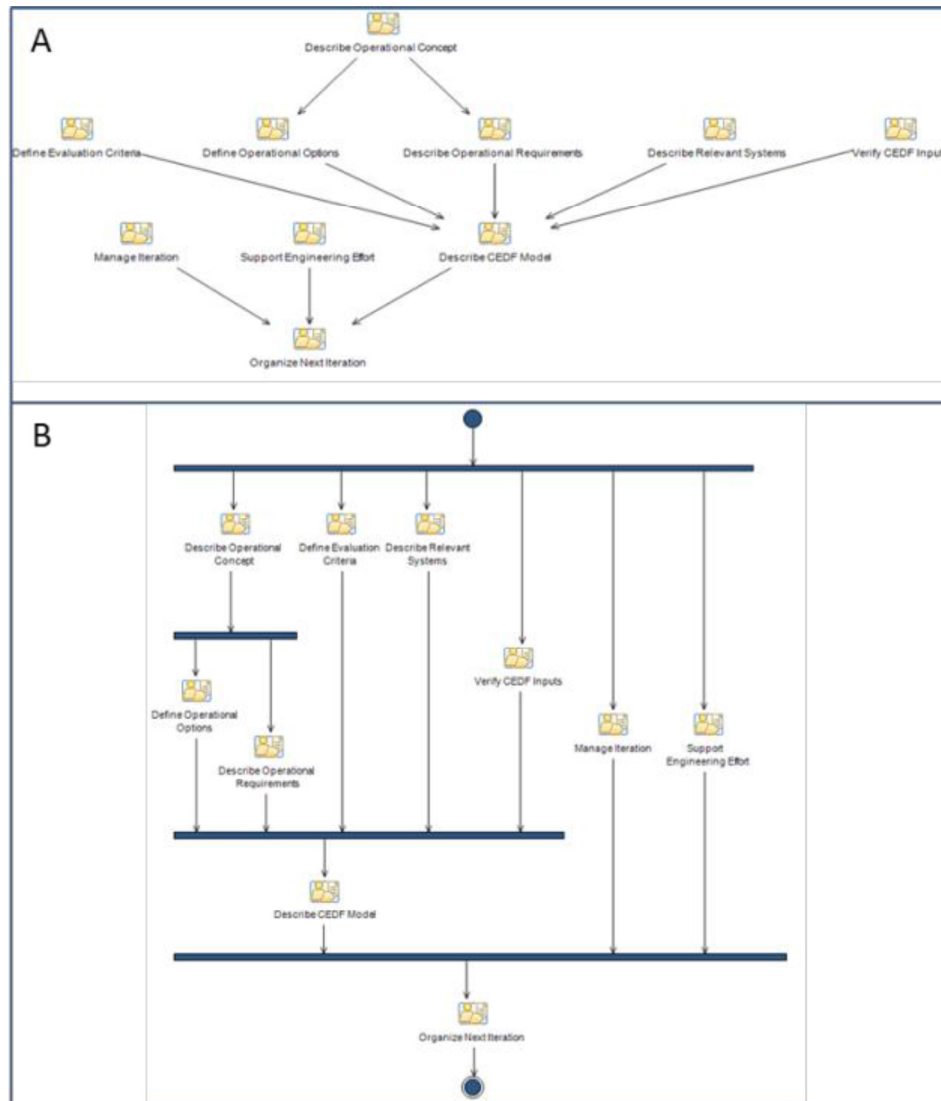


Figure 13: Generated (A) and Reviewed (B) Workflows for an Iteration

The construction of Comprehension and Elaboration stages within the Work Breakdown Structure tab consists essentially in including Capability Patterns representing iterations into the stage. For the Comprehension stage, it involves Comprehension 1 and Comprehension 2 iterations while it is Elaboration 1, Elaboration –n and Elaboration –final iterations for Elaboration. This inclusion is done by:

1. Adding each relevant iteration into the stage: This is done by selecting Apply Pattern/Extend from the contextual menu on the stage line. The selected capability pattern represents one of the defined iterations.
2. Checking the appropriate boxes for each iteration: The framework uses Planned, Repeat or Ongoing.







3. Revising the index and predecessors of each iteration: This represents the sequence of iterations within the stage.
4. Generating the workflow diagram and adjust it if required: This is done by selecting Open Activity Diagram from the contextual menu on the stage line.

#### 4.1.6 Publication rules

Once method content and process content are described, the publication is performed in three steps. The first step consists in customizing the navigation menu. The second validates the configuration. The final step publishes the Web documentation using the correct parameter set. Each step is further explained in the next paragraphs. Before starting publication, it is good practice to validate the Method Library (section 4.1.7).

1. Customize the navigation menu: Customization requires the creation of two custom categories in the current pattern plug-in: `delivery_process_category` and `navigation_view_generic`. These categories customize the information of the current categories defined in the `core.common` plug-in by replacing or extending part of their contents. Table 8: Modifications to Custom Categories for Navigation Menu Customization explains the changes to each category. The second line presents the goal of the modification. The third line explains how to apply the variability content. Finally, the last line illustrates the fields to be reviewed. Except for their name, no further information than that presented in the table needs to be added to each category.

Table 8: Modifications to Custom Categories for Navigation Menu Customization

Custom Category	delivery_process_category	navigation_view_generic
Goal	Add a menu for each stage within the Workflow menu	Indicate a relevant name for the process to be published.
Content Variability	<div> <div>▼ Content Variability</div> <div>Specify how this custom category relates to another custom category.</div> <div>           Variability type: <span>Extends and Replaces</span> </div> <div>           Base: <span> delivery_process_list</span> </div> </div>	<div> <div>▼ Content Variability</div> <div>Specify how this custom category relates to another custom category.</div> <div>           Variability type: <span>Extends and Replaces</span> </div> <div>           Base: <span> navigation_view_generic, core.common/Category</span> </div> </div>
Adjustments	Addition of each stage description in the Assign Tab. <div>  Inception, pattern.generic            Comprehension, pattern.generic            Elaboration, pattern.generic            Completion, pattern.generic         </div>	Change Presentation Name to specify a title for the process to be published. This name appears on top of the navigation menu. <div>             Presentation name: <span>CEP 2011 - Generic</span> </div>

2. Complete Method Configuration: Configuration involves the selection of plug-ins and views relevant to the process to be published. When the framework is respected, there should be no more than three plug-ins selected: `core.common`, `pattern.generic` and, for CEP adaption,

another pattern plug-in. Only one view needs to be specified in the framework: `navigation_view_generic`. Table 9 shows a sample of the information required for each tab of the Configuration editor. Each configuration will be similar from one process to another. The difference will lie in its Name, Brief description and the plug-ins selection.

Table 9: Information for Configuration Editor

Description tab	Plug-in and Package Selection tab	Views tab
<div> <div>▼ General Information</div> <div>Provide general information about</div> <div>Name: CEP 2011</div> <div>Presentation name:</div> <div>Brief description: Generic C</div> </div>	<div>Content:</div> <div> <div> <input checked="" type="checkbox"/> <input type="checkbox"/> core </div> <div> <input checked="" type="checkbox"/> <input type="checkbox"/> common </div> <div> <input checked="" type="checkbox"/> <input type="checkbox"/> pattern </div> <div> <input type="checkbox"/> <input type="checkbox"/> earlyCloseOut </div> <div> <input type="checkbox"/> <input type="checkbox"/> EM_Lite </div> <div> <input type="checkbox"/> <input type="checkbox"/> EM_Reuse </div> <div> <input type="checkbox"/> <input type="checkbox"/> FDO_Lite </div> <div> <input type="checkbox"/> <input type="checkbox"/> FDO_Reuse </div> <div> <input checked="" type="checkbox"/> <input type="checkbox"/> generic </div> </div>	<div>navigation_view_generic</div> <div> <div>CEP Overview</div> <div>Workflow</div> <div>Capability Engineering Team</div> <div>CEP Deliverables</div> <div>CEP Artefacts</div> <div>Example</div> <div>Glossary/Acronyms</div> <div>About</div> </div>

3. Publish the process: Selecting the Configuration/Publish menu launches the Publish Method Configuration wizard. To set the publishing parameters, the wizard requests the entry of information in four pages, as follows:
  - a. Select method configuration: Aladin lists the method configurations to choose from.
  - b. Select method configuration content: Check the radio button to Publish the entire configuration.
  - c. Select publishing options: Complete this page as shown in Figure 14: Publishing Options. Red-framed boxes indicate mandatory fields while orange-framed boxes indicate recommended fields.
    - Title: This corresponds to the title shown by the browser
    - Publish glossary: When this option is On, Aladin builds a glossary available on the top right corner of the site. This option provides a second path to reach the CEP glossary and acronyms as they are also available from the navigation menu.
    - Banner image: This field indicates an image file to serve as banner. When the field is empty, Aladin displays a default banner. The dimension of the default banner is  $1600 \times 67$ . The customized banner could be shorter (less than 1600) but the height should be the same for page length harmonization.
    - Diagrams: Check Publish activity detail diagrams that have not been manually created. This option is useful to insure all diagrams exist. However, it is preferable to edit them manually to validate their quality.
    - Layout: Check Include Method Content in descriptor pages. This option insures that method content information is integrated into the workflows. Select Work

Breakdown Structure as the default activity page in workflows. The framework does not provide any relevant information in the other tabs.

- Select destination directory and website format: Enter the destination directory and check the radio button for Static website.

**Publish Method Configuration**

**Select publishing options**

Select the publishing options. These options will be used to customize the look and behavior of the published website.

**Title and links**

Title: CEP 2011 - Generic (from title in wixard)

About content: [Select...]

Feedback URL: [ ]

**Glossary and index**

☒ Publish glossary ☐ Publish index

**Look and feel**

Banner image: C:\Aladin\Method Libraries\CEP2011\PACEM\_banner2.jpg [Select...]

**Validation**

☐ Check external hyperlinks ☐ Convert broken hyperlinks to plain text

**Diagrams**

☒ Publish activity detail diagrams that have not been manually created

☐ Publish activity diagrams for unmodified activity extensions

**Layout**

☐ Show relationship sub-folders in navigation trees

☐ Show related elements for roles, tasks and work products in navigation trees

☐ Show task descriptors in navigation trees

☒ Include method content in descriptor pages

☐ Publish process usage in role, task and work product pages linking to related descriptors

☐ Show all indirect (green) occurrences in extended patterns

Default tab for activity pages: Work Breakdown Structure

< Back Next > Finish Cancel

Figure 14: Publishing Options

Once the parameters are set, click Finish and the Web documentation process will generate. It is a good practice to verify the resulting website (see section 4.1.7).

#### 4.1.7 Validation checklist

The validation occurs at two different times. The first validation is done prior to publication and focuses on method content, process content and configuration. The second validation concentrates

on the website resulting from the publishing process. This section provides two main checklists: one to use before publishing and another to use after publishing.

### **Before publishing**

The core.common plug-in is reviewed.

1. All method elements are created in the right content package.
  - ♦ The ACG\_description, AFDO\_description, DOA\_description, DSoSA\_description and MEE\_description content packages contain their complete set of tasks, work products and guidance.
  - ♦ All acronyms and definitions are defined as term definition guidance and located respectively in the CEP\_Acronyms\_description and CEP\_Glossary\_description content packages.
  - ♦ CEP overview and stages documentation are described as supporting material guidance and located respectively in the CEP\_Overview\_description and CEP\_Stage\_description content packages.
  - ♦ Each CEP generic role is defined under the CEP\_Roles\_description content package.
  - ♦ The Common\_Work\_Products\_description content package contains all work products that are not specific to a discipline.
  - ♦ The Generic pages used in the website (About, Home, Copyright, etc.) are defined in Default\_Publication\_Package.
2. All mandatory fields of each method element shown in Table 10 are completed with respect to the framework rules.
3. All method elements have been previewed:
  - ♦ Their content is complete and respects the writing conventions (section 4.1.2).
  - ♦ The copyright is displayed at the bottom of the page.
  - ♦ Their formatting is coherent and homogeneous.
  - ♦ Their images are shown correctly.
  - ♦ There is a reference in the text for each guidance added to the method element.
4. Each standard category is completed:
  - ♦ Each task defined in Method Content is included in a Discipline category.
  - ♦ Each artifact defined in Method Content is included in a Domain category.
  - ♦ Each role defined in Method Content is included in one Role Set.
5. Each custom category presented in Figure 7 is defined.



Table 10: Summary of Mandatory Fields for Method Elements

Guidance	Work Product	Roles	Tasks
<ul style="list-style-type: none"> <li>• Name (all types of guidance)</li> <li>• Presentation Name (all types of guidance)</li> <li>• Brief Description (all types of guidance)</li> <li>• Check Items (checklist type of guidance)</li> </ul>	<ul style="list-style-type: none"> <li>• Name (artifact and deliverable)</li> <li>• Presentation Name (artifact and deliverable)</li> <li>• Brief Description (artifact and deliverable)</li> <li>• Domains (artifact)</li> <li>• Deliverable Parts (deliverable)</li> </ul>	<ul style="list-style-type: none"> <li>• Name</li> <li>• Presentation Name</li> <li>• Brief Description</li> <li>• Responsible for Role Sets</li> </ul>	<ul style="list-style-type: none"> <li>• Name</li> <li>• Presentation Name</li> <li>• Brief Description</li> <li>• Purpose</li> <li>• Steps</li> <li>• Primary Performers</li> <li>• Mandatory Inputs</li> <li>• Outputs</li> <li>• Disciplines</li> </ul>

The pattern.generic plug-in is reviewed.

1. All activities are created in the right Process Package.
  - ♦ ACG, AFDO, DOA, DSoSa and MEE Process Packages contain activities specific to their respective discipline.
2. All activities are described and are up to date.
  - ♦ Each activity includes only tasks from their respective discipline.
  - ♦ Their work breakdown structure is defined. Their tasks are added, characterized and sequenced.
  - ♦ Synchronization with source property is checked for each task included in the activity.
  - ♦ Their content is synchronized with method content.
  - ♦ Their activity detail diagram is generated and reviewed.
3. All iterations (comprehension 1, comprehension 2, elaboration 1, elaboration-n and elaboration-final) and stages without iteration (inception and completion) are described.
  - ♦ Their work breakdown structure is defined. Their activities are added, characterized and sequenced.
  - ♦ The tasks in an activity not relevant to an iteration or stage are suppressed.
  - ♦ The activity diagram is generated and reviewed.



4. The comprehension and elaboration stages are described.
  - ♦ Their work breakdown structure is defined. Their iterations are added, characterized and sequenced.
  - ♦ The activity diagram is generated and reviewed.
5. The custom categories to update the navigation menu are described.
  - ♦ `delivery_process_category` extends and replaces the `delivery_process_list` in `core.common` and includes stages defined in the `stages-iterations` process package.
  - ♦ `navigation_view_generic` extends and replaces the `navigation_view_generic` in `core.common` with a specific presentation name.

The configuration is reviewed:

1. The configuration is created and includes only the relevant plug-ins.
2. The configuration identifies a view.

## **Website**

1. The banner is correctly displayed: It is the correct banner with the correct resolution and positioning.
2. The correct title appears in the browser and is spelled correctly (spelling, spacing and proper case): This corresponds to the title in publishing options dialog box.
3. The title above the navigation menu is correct (name, spelling, spacing and proper case): This corresponds to Presentation Name in the `navigation_view_category`.
4. All navigation menus are shown and their tree structure is correct.
5. All pages are correctly displayed: Each completed section appears with text correctly formatted. Image quality is good. Workflow pictures are well formed. The copyright is shown at the page bottom except on Workflow pages.
6. All links work properly: Text and image links lead to the correct pages.
7. Glossary at the top right corner works properly: The glossary is available from that menu only when Publish Glossary is checked in the publishing options.
8. About at the top right corner works properly.

This section summarizes the rules, formats and tools prescribed by the framework to document a generic CEP within Aladin. It focuses on what has to be entered and how it should be completed. It does not provide details on the mechanics and user instructions for Aladin. This information can be found in [25]. The generation of CEP adaptation follows the same rules as those presented

for the generic CEP with some additions and uses the content variability functionalities more intensively.

## 4.2 CEP adaptation

The CEP Documentation Framework allows the documentation of CEP adaptations through the use of process patterns. The use of pattern aims to optimize reusability within the framework. Only new or changed elements are replicated within the pattern documentation, and all common elements will be referenced. In addition to the various method components described before, the process pattern textual description is described in a specific template, which is used by Aladin to provide a search feature. As for the CEP generic documentation, a logical sequence is recommended to populate the framework properly. The following paragraphs will first explain how to properly fill a process pattern template. The logical sequence for documenting the framework is then exposed, as is guidance, rules and validation checklists to populate and publish the adapted CEP website. To illustrate this section, examples are taken from the Hi-Spin pattern. This pattern mimics the SCRUM method 24 and applies its agile principles to the CEP. A complete illustration and step-by-step implementation of the High-Spin pattern is found in Annex C.

Almost all the rules and guidance explained in section 4.1 apply here; however, some specificity can be found in the pattern documentation, the strategy of implementation and some other rules.

### 4.2.1 Basic structure creation

Each pattern will be documented within its own plug-in. Therefore, it is mandatory to create this plug-in for each pattern. The plug-in will be named “pattern.name-of-the-pattern” (example pattern.high-spin). This step creates the basic repository structure needed to document the pattern.

Within the plug-in, the process pattern has to be created within the process/process pattern part of the plug-in, as shown in Figure 15.

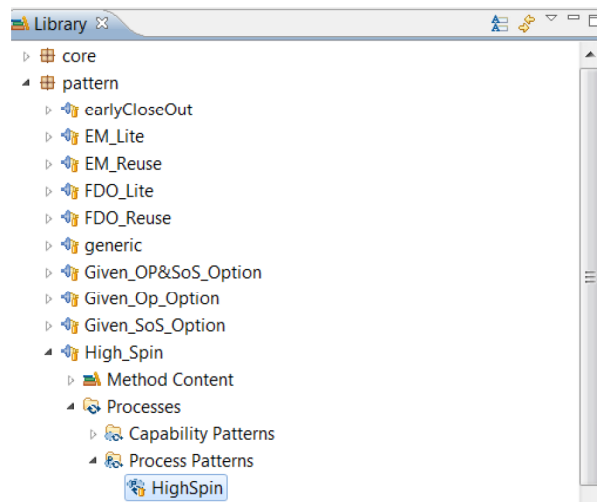


Figure 15: Process Pattern Location

## 4.2.2 Pattern description

After the basic structure creation, the first task is to literally document the process pattern using a process pattern template. The template consists of the content tab of the process pattern editor. It includes five different sections: General Information; Context Information; Solution Information; Version Information and Configuration. The last two sections are generic to all Aladin elements and will not be discussed here.

### 4.2.2.1 General information

The General Information section lists the name, presentation name, brief description and purpose of the pattern, as shown in Figure 16 below. The last field, Related Patterns, allows the user to select relevant patterns within the repository.

The screenshot shows a software window titled "HighSpin" with a sub-header "Process Pattern: HighSpin". The main content area is divided into sections. The "General Information" section is expanded, showing fields for "Name" (HighSpin), "Presentation name" (HighSpin), "Brief description" (Instead of planning stages, this pattern plans sprints in the Inception stage. It manages all following stages in rapid scrums or time boxes called "sprints". The validation is done through each sprint.), and "Purpose" (To present the client with a build up of the solution in rapid succession so that he/she can judge the solution and also better formulate his/her needs and goals.). Below these is a "Related patterns" section with an empty list and buttons for "Add...", "Edit...", and "Remove". At the bottom, there are tabs for "Context Information", "Solution Information", "Version Information", and "Configuration". A footer bar contains tabs for "Description", "Work Breakdown Structure", "Team Allocation", "Work Product Usage", and "Consolidated View".

*Figure 16: Process Pattern Template – General Information Section*

- Name: In this internal field, provide a significant name for the pattern.
- Presentation name: In this published field, provide the pattern name as it should appear in the Web documentation.
- Brief description: In this published field, provide one or two sentences describing what the pattern is. The description should not include any directives on how to perform the process.

- **Purpose:** In this published field, provide a short sentence starting with “to” that indicates the goal of the pattern. Example: To present clients with a build up of the solution in rapid succession so that they can judge the solution and also better formulate their needs and goals.
- **Related Patterns:** In this published field, provide a short sentence starting with “to” that indicates the goal of the task, e.g. To determine a validated set of operational and performance parameters criteria.

#### **4.2.2.2 Context information**

The Context Information section documents the situation in which the pattern is relevant. This section is used to screen available patterns in order to present to the user the patterns which are applicable to the CEP initiative having the same particularities. An example is shown in Figure 17.

A mandatory rule here is to fulfill only elements specific to the pattern. For example, if a pattern is applicable regardless of the size of the CET Team, then the Size of team field should be left as “Not Applicable” or “Not Relevant.”

- **Problem description:** Explain the particularity of the project’s environment and the difficulties being addressed by the pattern.
- **Affected Discipline:** Indicate the scope of the pattern, i.e. what disciplines of the method are modified by this particular pattern.
- **Affected Stages and Iterations:** Indicate the scope of the pattern, but related to the parts of the process affected.
- **Team Characteristics:** Select the characteristics of the designer’s team applicable to this particular pattern, such as its size, level of expertise and cohesion.
- **CET members’ availability:** Select the level of availability of CET members required by the patterns.
- **Initiative characteristics:** Select the particularity in time, complexity, budget, etc., for initiatives that could use this pattern.
- **Deliverable particularities:** For each standard CEP deliverable, select whether the initiative requires a rapid delivery of the deliverable or if the initiative doesn’t need the deliverable.

HighSpin

## Process Pattern: HighSpin

General Information

Context Information

Provide context information about this process pattern.

Problem Description: A CE Initiative might use the High Spin pattern when:

- The client has difficulty expressing the capability gap;
- The Force Development Option has to be demonstrated rapidly;
- A knowledgeable client representative is available on site who can make decisions;
- The priorities in solving capability gaps shift rapidly;

Affected Disciplines

☒ Manage Engineering Effort
 ☒ Analyze Capability Gap
 ☒ Develop Operational Architecture
 ☒ Develop SoS Architecture
 ☒ Asses Force Development Options
 ☒ All

Affected Stages and Iterations

☒ Inception
 ☒ Comprehension Iteration 1
 ☒ Comprehension Iteration 2
 ☒ Elaboration Iteration 1
 ☒ Elaboration Iteration 2
 ☒ Elaboration Iteration 3
 ☒ Completion
 ☒ All

Team Characteristics

Size: 
 Distributed: 
 Cohesion: 
 CEP Knowledge: 
 Gap Knowledge:

CET Members Availability

CET Lead: 
 Op. Architect: 
 FDO Analyst: 
 Arch. Modeler: 
 OR Analyst: 
 CET Coordinator: 
 SoS Architect: 
 IT Specialist: 
 Req. Analyst:

Stakeholders Availability

Capability Manager: 
 PRICIE Reps: 
 Operational Reps: 
 External Users:

Initiative Characteristics

Timeframe: 
 Budget: 
 Gap Complexity: 
 Gap Novelty: 
 Data Classification: 
 Existing IT Support:

☐ This pattern is applicable to an ongoing CE initiative  
☒ This pattern can be partially applied

Deliverables

Engineering Management Plan: 
 Strategic Context Analysis: 
 Requirements: 
 Operational Architecture: 
 SoS Architecture: 
 Force Development Options:

Figure 17: The Context Information Section

#### 4.2.2.3 Solution information

This section documents the solution part of the pattern.

- Forces in presence (shown in Figure 18): Indicate here the different forces that influence the CEP initiative. These forces will balance the degree of application of the measures suggested

within the pattern. They will dictate an equilibrium within the measures proposed in the pattern, according to the actual CEP initiative context. For example, the low availability of resources is a force pushing toward the use of a reduced team. On the contrary, the complexity of the gap might influence the use of more designers. Those forces are shaping the initiative context in opposite ways. The pattern takes these forces into account and its solution will propose a solution in between the opposite courses of action.

- Solution description: Describes a general description of the pattern, i.e. what changes are applied compared to the regular process. The detail of the method and process workflows will be documented within the pattern plug-in, as detailed in the corresponding sections below.
- Resulting context: Describe the context of the initiative after the application of the pattern. This description should show what has been solved, as well as the remaining problems and the potentially new challenges created by applying the pattern.
- Rationale: Here, the user can justify the potential efficiency of the pattern.
- Known use: Indicate known applications of the pattern.
- Reference. If applicable, put references in this field.



**Process Pattern: HighSpin**

► General Information

► Context Information

▼ Solution Information

Provide solution information about this process pattern.

Forces in presence:	<p>The influences that are in favor of the pattern are:</p> <ul style="list-style-type: none"> <li>• Lack of well-defined gap or goal;</li> <li>• Multiplicity of functionalities that can be developed independently;</li> <li>• The time-boxed nature of the planned CE Initiative;</li> <li>• The availability of knowledgeable client representative on the CET site who can make decisions for</li> </ul>
Solution Description:	<p>This pattern is a rapid turn around capability gap recommendation method useful when the capability gap is fluid. Effort is dispensed in short bursts. The decision to proceed with the recommendation or to proceed through a second, or nth, sprint can then be made on the basis of the recommendation and additional information that might have come up during the sprint. The pattern adds velocity to nominal CEP. It minimizes effort spent on investigating improbable capability options.</p>
Resulting context:	<p>The application of the High Spin pattern should achieve the following benefits:</p> <ul style="list-style-type: none"> <li>• Higher quality recommendation to client for a fluid capability gap;</li> <li>• Potential reduction in EM effort and costs;</li> <li>• Potential reduction of the various CET roles involvement.</li> </ul>
Rationale:	<p>The pattern works because it provides a gradual improvement in the quality of the recommendation while minimizing effort spent on investigating improbable capability gaps. It works because there is a close collaboration with the client to refine priorities and decision criteria. This pattern is a rapid turn around capability gap recommendation method useful when the capability gap is fluid. Effort is dispensed in short bursts called "sprints". The decision to proceed with the recommendation or to proceed through a second or nth sprint can then be made on the basis of the recommendation and additional information</p>
Known uses:	<p>None at present time</p>
References:	

► Version Information

► Configuration

Figure 18: The Solution Information Section

### 4.2.3 Completion sequence

Once the basic structure was created and the adapted CEP was described textually within a process pattern template, its components still remain to be documented within the pattern plug-in. The completion sequence consists of five major activities:

- Identification of needed changes and the establishment of an implementation strategy;
- Creation of the necessary framework structure (configuration and plug-in);
- Documentation of new and modified method elements;
- Documentation of new and modified process elements; and

- Final validation and publication of the new CEP documentation.
1. Strategy of implementation
    - a. Identify changes and differences between the adapted and generic CEP (e.g. stages, activities, tasks, steps, roles, artifacts, deliverables, guidance).
    - b. Establish the strategy for implementation. Determine how and where modified or new elements will be documented (for instance, are individual activities being replicated or will it be the whole iteration or stage?),
  2. Creation of the framework structure
    - a. Create a configuration with a significant name and select the core.common and pattern.generic as well as the newly created plug-in (e.g. configuration CEP High-Spin 2011 refers to the core.common, pattern.generic and pattern.highspin plug-ins).
    - b. Create a process pattern in the process content of the plug-in. Document each of its properties as described in section 4.2.2.
    - c. Create the needed structure for the method content, replicating the core.common structure only for new or modified method elements (see Table 1 and Table 2).
    - d. Create the needed structure for the modified or new process content, replicating the cep.generic structure to accommodate new or modified activities, stages or iteration content (see Table 3).
  3. Creation and modification of method elements
    - a. Create method elements for each new or modified element in the appropriate structure as in 2.c. As per section 4.1., the sequence of creation should be:
      1. Guidance;
      2. Artifacts;
      3. Deliverables;
      4. Roles; and
      5. Tasks.
    - b. Document the new elements with respect to all rules from section 4.1.
    - c. Document the modified elements with respect to the “content variability” rules from section 4.2.5.
  4. Creation or modification of process elements (activities and stage/iteration)

- a. Create a process package structure mimicking the pattern.generic structure to insert the new or modified activities (e.g HS\_MEE for new or modified management activities). The stages/iterations process package is almost always recreated. If needed, create stages/iterations according to the chosen strategy of implementation.
- b. Create each new or modified activity in the appropriate process package.
  - i. Refer to the variability rules for modified activities.
- c. Create the work breakdown structure in the Work breakdown structure tab of the Process pattern (in the processes section of the plug-in). Activities and capability patterns will be selected either from the pattern.generic plug-in or the new process pattern plug-in. They will be included with the Extend option in order to automatically include future changes in the base activities.
- d. Adjust the scheduling of activities, use the activity diagram and place arrows to identify the activity dependencies. The work breakdown structure will be automatically adjusted.

## 5. Validation and publishing of the Web documentation

- a. Validate all method elements as variability and replication mechanisms may have side effects.
- b. Customize the navigation menu as in 4.1.6.
- c. Publish the adapted CEP website.

### 4.2.4 General process pattern rules

- A CEP adaptation scope is a complete delivery process (life cycle).
- A pattern plug-in is created for each pattern. It is named pattern.name-of-pattern. The use of a dot in the name allows all pattern plug-ins to be grouped together.
- For optimized reusability, only new or changed content is created within the pattern plug-in. Untouched content will be referenced through its original plug-in, by way of the deep copy mechanism.

### 4.2.5 Method elements rules

As documented in 4.1.3, method elements are documented through their specific editor. Only new and modified elements will be documented in the pattern plug-in.

To modify an element, for example a task, it is necessary to recreate within the pattern plug-in a new element with the same name as the original one. Its relation with the original task will be indicated in the content variability section of the description tab. There are four variability rules as detailed below:

- Variability rules:
  - The contribute variability association will conserve all existing field values of the original entity. It will add the content of the contributing one at the end of the relevant attributes. The relationships from the contributing element are added to the base elements.
  - In the extend variability, field values and associations are inherited from the base element. The attributes documented within the extend element will replace the ones from the base element. Undocumented attributes will be replaced by the content of the base element.
  - The replace variability association replaces a base element with a new one. All field values will replace the base element field values. All original content will be lost.
  - The extend and replace variability association is a combination of the two preceding ones. It will replace the values that have been redefined and leave the others untouched. Only the field values that have been documented in the new elements are replaced. The original text is lost for these attributes.
- Table 11 shows the results of the variability association applied to two method elements. We selected two fields of an original method element (second column) and created a new method element with a variability relationship toward the original one (third column).
- The resulting values are indicated in the last column. The variability mechanisms also have similar effects on linked items:
  - Extend: All relationships (role-tasks, tool-artifacts, etc.) have to be re-specified as they are lost with this variability setting.
  - Replace: The links with other elements (tasks, etc.) are kept. Useful to replace an element with the same name.
  - Extend and Replace: All links between elements are lost.
  - Contributes: All original content is kept, as well as links with original links.
- For new elements, each task must be assigned to a discipline; each role must be assigned to a role set and each work product must be assigned to a domain.
- If a new role contributes to a task, the task has to be duplicated in the pattern plug-in (with a contribute variability) and the association between the role and the task has to be specified in the Roles tab of the task editor.
- If a new role is responsible for a work product, the Responsible field from the Work Products tab in the role editor needs to be completed.

Table 11: Variability Mechanisms

Variability association	Original fields	New values	Results
<b>Contribute</b>	<b>Purpose:</b> This is the original purpose.	<b>Purpose:</b> A modified purpose	<b>Purpose:</b> This is the original purpose. A modified purpose.
	<b>Brief Description:</b> This is the original Brief Description.	<b>Brief Description:</b>	<b>Brief Description:</b> This is the original Brief Description.
<b>Extend</b>	<b>Purpose:</b> This is the original purpose.	<b>Purpose:</b> A modified purpose	<b>Purpose:</b> A modified purpose
	<b>Brief Description:</b> This is the original Brief Description.	<b>Brief Description:</b>	<b>Brief Description:</b> This is the original Brief Description.
<b>Replace</b>	<b>Purpose:</b> This is the original purpose.	<b>Purpose:</b> A modified purpose	<b>Purpose:</b> A modified purpose
	<b>Brief Description:</b> This is the original Brief Description.	<b>Brief Description:</b>	<b>Brief Description:</b>
<b>Extend and replace</b>	<b>Purpose:</b> This is the original purpose.	<b>Purpose:</b> A modified purpose	<b>Purpose:</b> A modified purpose
	<b>Brief Description:</b> This is the original Brief Description.	<b>Brief Description:</b>	<b>Brief Description:</b> This is the original Brief Description.

#### 4.2.6 Activity/iteration/stage/delivery process rules

The framework uses capability patterns to document activities, stages and iterations, as explained in 4.1.4 and 4.1.5. Only modified and new elements are documented within the pattern and the same content variability rules used for the Method Content apply here.

The selection of elements for building activities, iterations and stages is made within the pattern.generic and the current pattern plug-ins.

The delivery process workflow is built in the work breakdown structure tab of the process pattern element. The constituting stages are selected either within pattern.generic or within its own plug-in.

See Annex C for a complete illustrated example.



#### **4.2.7 Validation checklist**

In addition to the checklist presented in section 4.1.7, the following items have to be checked:

- That each new role is associated to a role set.
- That each new task is associated to a discipline.
- That each new work product is associated to a domain.
- That the list of work products that each new role is responsible for is validated.
- That the links of each modified object are validated as the variability mechanism used may have affect these links.

#### **4.2.8 Publication**

The regular publication process is used for a patterned CEP. However, some minor adjustments have to be made:

- Configuration: It is important to select the `core.common` and `pattern.generic` plug-ins in addition to the current pattern plug-in within the configuration.
- The custom categories of the `pattern.generic` must be removed from the package selection, as shown in Figure 28 in Annex C.



## 5 Conclusion

---

Since the creation of the capability engineering process within the CapDEM project, different adaptations of the process were needed for a better suitability to some projects. The PACEM project introduced the concept of process patterns in order to ease the adaptation of engineering processes and augment the reusability of such modifications. This technical memorandum describes in detail the documentation framework used to document and publish capability engineering processes through the use of process patterns.

Chapter 2 introduced different concepts used within this document. Agility, process and process patterns are the key parts of the framework structure. Method engineering provides the foundation for the metamodel. Different standards and metamodels, such as UMA and SPEM, are part of the framework structure. Aladin is the tool based on the open source Eclipse Process Framework built to implement the framework. It provides the documentation, storage and publication features.

Chapter 3 detailed the structure used to document CEP, as well as the rules and principles governing the method, process and publishing elements. The framework is organized through plug-ins, methods and process packages. Rules and conventions are proposed to ease the framework maintenance and integrity. The organization of the framework in different plug-ins, packages and configurations is explained here.

Chapter 4 illustrated the use of the framework. The elements of the generic capability engineering framework are first introduced, followed by the process pattern template. The construction of a generic CEP is first explained, followed by the steps needed to implement a process pattern for a custom-fit CEP. Annex C contains a step-by-step detailed illustration of the implementation of a process pattern to provide a concrete example for the reader.

All the previous elements constitute a versatile toolset for the method engineer or project manager willing to document and easily publish an adapted engineering process of any scope. Such a framework leads the way toward process instrumentation and enactment through workflow and monitoring tools.

## References

---

- [1] Lizotte, M., C. Lalancette, and C. Nécaille. 2006. Capability Engineering for Strategic Decision Making. In Proceedings of the 16th Annual International Symposium of the International Council on System Engineering (Orlando, FL). Seattle: INCOSE.
- [2] Lizotte, M., C. Nécaille, C. Lalancette, and G. Dussault. 2009. Capability Engineering Process Version 2008 (CEP 2008) (U), (DRDC Valcartier TR2008-266). Defence R&D Canada – Valcartier.
- [3] Nécaille, C. 2007. Canada National Update. Internal publication (DRDC Valcartier SL-2007-148). Defence R&D Canada – Valcartier.
- [4] Nécaille C., 2009. Capability Engineering Process (CEP) and WSAF, Overview, status and perspectives (U). Internal Publication (DRDC Valcartier SL 2009-314). Defence R&D Canada – Valcartier.
- [5] Nécaille, C. 2011. Process Patterns for Agile Capability Engineering Methodology: The PACEM Project. In Proceedings of the 21st Annual International Symposium of the International Council on System Engineering (Denver, CO). Seattle: INCOSE.
- [6] Mackley, T., S. Barker, and P. John. 2008. Concepts of Agility in Network Enabled Capability. In RNEC Conference 2008. Leeds, UK.
- [7] Kumar, K. and R. J. Welke. 1992. Method Engineering, A Proposal for Situation-Specific Methodology Construction. In Systems Analysis and Design: A Research Agenda, Cotterman and Senn (eds), Wiley. Pp. 257-268, 1992.
- [8] Henderson-Sellers, B. and J. Ralyté. 2010. Situational Method Engineering: State-of-the-Art Review, in *Journal of Universal Computer science* 16, 3: 424-478.
- [9] OMG. 2008. SPEM Software Process Engineering Metamodel Specification, Version 2.0.
- [10] ISO/IEC. 2007. ISO/IEC 24744: Software Engineering – Metamodel for Development Methodologies. International Organization for Standardization/International Electrotechnical Commission, Geneva.
- [11] Rolland, C. 1994. Modeling the Requirements Engineering Process, Information Modelling and Knowledge Bases V: Principles and Formal Techniques 5, 85.
- [12] Barter, R. 1998. A Systems Engineering Pattern Language. In Proceedings of the 8th Annual International Symposium of the International Council on Systems Engineering (Vancouver, BC). Seattle: INCOSE.
- [13] Haumer, P. 2006. Increasing Development Knowledge with EPFC, Eclipse Review, Spring 2006, 2006.

- [14] Fredrickson, J. T. and P. Haumer. 2007. Unified Method Architecture. Patent 11/238500 International Business Machines Corporation.
- [15] Alexander, C. 1977. The Timeless Way of Building. New York: Oxford University Press.
- [16] Gamma, E., R. Helm, R. Johnson, and J. Vlissides. 1995. Design Patterns: Elements of Reusable Object-Oriented Software. Reading, Mass.: Addison-Wesley.
- [17] Coplien, J. O. and D. C. Schmidt. 1995. Pattern Languages of Program Design. Reading, Mass.: Addison-Wesley.
- [18] Ambler, S. W. 1998. Process Patterns. Cambridge University Press.
- [19] Buschmann, F., R. Meunier, H. Rohner, P. Sommerlad, and M. Stal. 1996. Pattern-Oriented Software Architecture Volume 1: A System of Patterns. Wiley.
- [20] Haskins, C. 2005. Application of Patterns and Pattern Languages to Systems Engineering. In Proceedings of the 15th International Council on System Engineering International Annual Symposium (Rochester, NY). Seattle: INCOSE.
- [21] Cloutier, R. J. and D. Verma. 2007. Applying the Concept of Patterns to Systems Architecture. Systems Engineering 10, 2: 138-154.
- [22] Eclipse contributors et al. 2008. "Method Authoring Method for Eclipse Practice Library (EPL)," online, <http://epf.eclipse.org/wikis/mam/>, May 2012, 2008.
- [23] Eclipse contributors et al. 2011. "OpenUP library," online, <http://epf.eclipse.org/wikis/openup/>, May 2012, 2008.
- [24] Eclipse contributors et al. 2008. "SCRUM library," online, <http://epf.eclipse.org/wikis/scrum/>, May 2012, 2008.
- [25] Eclipse Process Framework (EPF) Composer, Installation, Introduction, Tutorial and Manual, online, [http://www.eclipse.org/epf/general/EPF\\_Installation\\_Tutorial\\_User\\_Manual.pdf](http://www.eclipse.org/epf/general/EPF_Installation_Tutorial_User_Manual.pdf), accessed May 2012.

This page intentionally left blank.

## Annex A Unified Method Architecture Metamodel

This annex is a short description of the Unified Method Architecture (UMA) metamodel. The main principles behind UMA are the separation between method and process elements and the late binding of method elements to their process counterparts. UMA defines the metamodel used to structure method content and process within EPF and Aladin. Concrete UMA model classes can be grouped into two broad categories and several sub-categories.

### A.1 Method content

The following UML class diagram shows the organization of the Method Content classes.

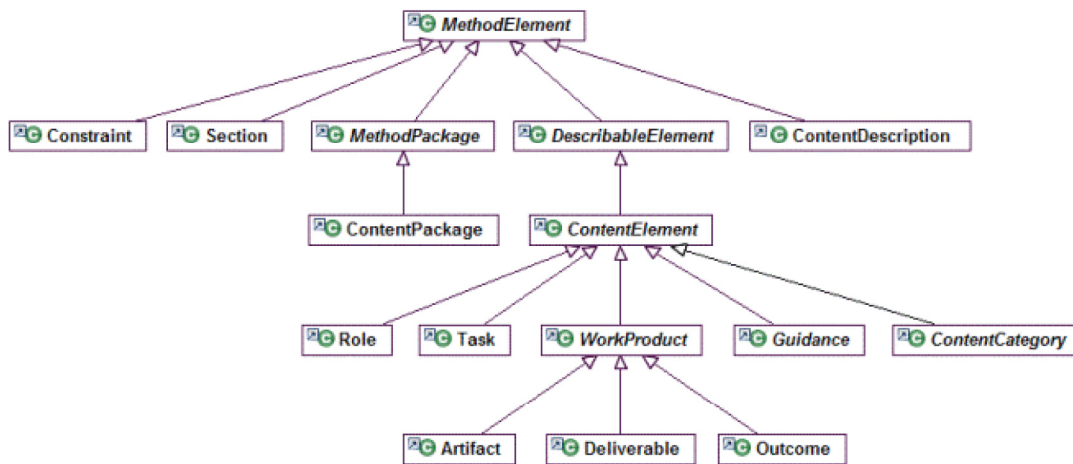


Figure 19: UMA content model

Method content can be sub-divided into the following categories:

- Core Method Content: role, task and work product (artifact, outcome and deliverable);
- Guidance: checklist, concept, example, guideline, estimation considerations, practice, report, reusable asset, roadmap, supporting material, template, term definition, tool mentor and whitepaper; and
- Content Category: discipline grouping, discipline, domain, work product, role set grouping, role set, tool and custom category.

### A.2 Process elements

The following UML class diagram shows the organization of the process element classes.

UMA defines a method library as a root container for all method elements. The following UML class diagram shows the organization of the key classes in a method library instance.

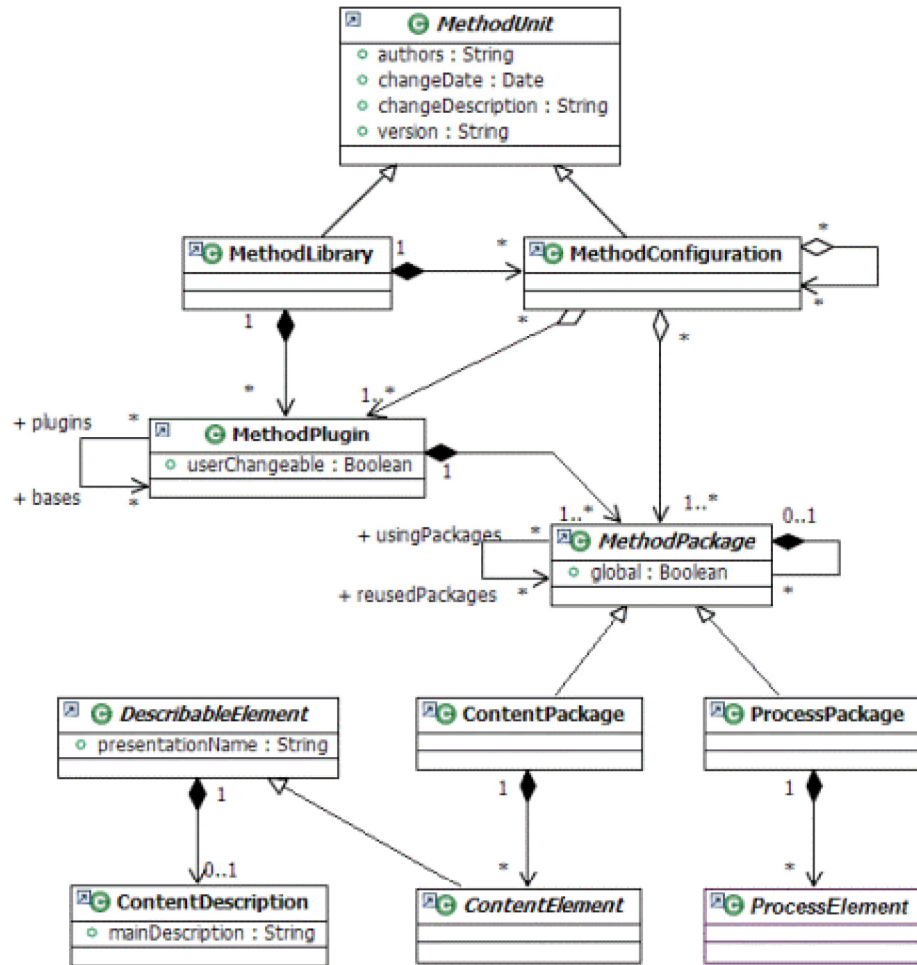


Figure 20: UMA Process Elements



## Annex B List of acceptable verbs

---

This annex reproduces a list of recommended verbs for the formulation and descriptions of tasks and activities in the table. This list comes from the Method Authoring Method (MAM) project [22].

Method authors may add verbs but they should do so in such a way as to avoid duplication. If a new verb is created, ensure that the definition of what it means is included in the configuration.

*Table 12: List of Acceptable Verbs*

Verb	Meaning	Comments
Acquire	To come into possession of	
Analyze	To determine the relationship of component parts	
Assemble	To fit parts together	Especially useful for deliverables
Assess	To make a judgment of worth	
Assign	To appoint to a post or duty	
Build	To construct by putting parts or materials together	
Capture	To document	
Categorize	To analyze and group according to a particular criteria	
Cleanse	To purify	
Communicate	To transmit information so that it is understood	
Conduct	To direct or manage	
Configure	To set up for operation	
Confirm	To verify that you have what's needed	
Detail	To provide the details for an outlined artifact	
Develop	To bring to maturity; to provide a more specific definition	
Elicit	To draw out or evoke	
Enable	To make operational	
Estimate	To judge approximate value	
Evaluate	To determine significance or worth	

Gather	To bring together into one collection	
Gather	To locate and bring together	
Identify	To establish identity	
Implement	To fulfill; to realize	
Initiate	To facilitate the beginning	
Install	To place in position of use	
Manage	To direct or supervise	
Obtain	To get or attain by planned action or effort	
Outline	To describe key elements	
Perform	To do	
Plan	To describe objectives, as well as a sequence and deadline for reaching a goal; to specify how to reach a goal	
Prepare	To make ready	
Prioritize	To set priorities	
Receive	To acquire, come into possession	
Reconcile	To check against another for accuracy and make them match	
Release	To make available for use	
Review	To examine carefully, looking for errors, omissions, ambiguity, inconsistency	
Run	To perform, generally by executing a program on a computer	
Select	To choose	
Specify	To name or state explicitly or in detail	
Ship	To transport an item	
Train	To teach a task or job	
Understand	To comprehend meaning	
Validate	To confirm that a solution or process is correct	
Verify	To ensure correctness according to specific criteria	

## **Annex C Step-by-step creation of the high-spin pattern**

---

This annex illustrates the documentation and publication of an adapted CEP, according to the guidance and rules established within the report.

As an example, the “High-Spin” process pattern was chosen. This pattern merges the scrum approach with the generic CEP process.

As described in the report, documentation of a process pattern is done in five stages: the identification of changes and strategy of implementation; the creation of the needed structure; the documentation of new and modified method elements; the documentation of new and modified process elements; and, the documentation of the navigation menus needed for the publication of the whole process pattern.

### **C.1 Strategy of implementation and identification of changes**

In summary, this pattern combines the comprehension and elaboration stages into a sprint. A sprint requires a notebook (new deliverable) to plan, close and manage it. The application of the High-Spin pattern involves close involvement of the client in many tasks.

It results in the following changes:

- Creation of the pattern.highspin plug-in and its corresponding high-spin configuration;
- Creation of the “High-Spin” Process pattern and documentation of the template;
- New role: Client Representative;
- New artifact: Sprint Notebook; which falls under the responsibility of the CET\_Lead\_Assistant;
- Modification to the CET lead assistant role for sprint notebook responsibility;
- Revisiting management tasks to refer to sprint and to include a new role or new artifact.
- Creation of a new stage “Sprint” and modification of the delivery process.

There is no change in the menu and navigation views of the website, except for minor modifications such as the title and copyright information.

The list of changes for each item are found after the relevant sub-sections.

### **C.2 Plug-in and configuration**

A “CEP High-Spin 2011” plug-in and a corresponding configuration are created. Each new element will be prefixed by “HS\_” for easier recognition, except for the process pattern below.

Within the process part of the plug-in, a process pattern is created and its template is documented as shown in Figure 21.

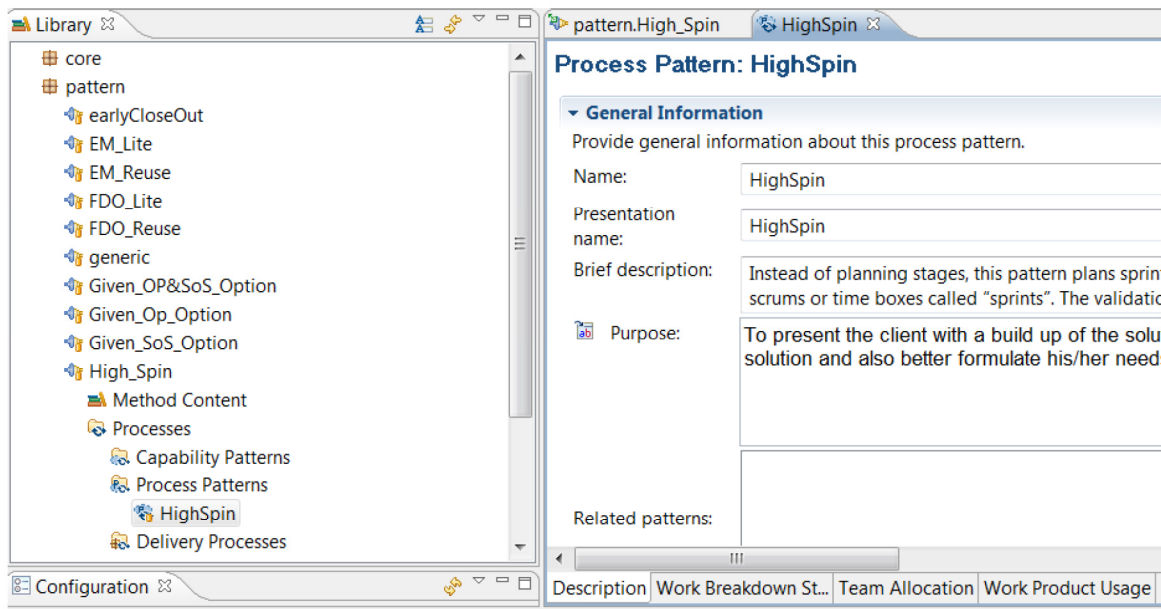


Figure 21: Process Pattern Documentation

Prior to the implementation of any changes, it is mandatory to select the plug-ins that will be used in the configuration to be able to reference the generic CEP elements. For high-spin, every core.common and pattern.generic is selected. The pattern.generic.custom category elements must be subtracted, as they contain publication menus which will be superseded by our own content (see Figure 22). Details of views associated with the pattern will be discussed in the Publication section.

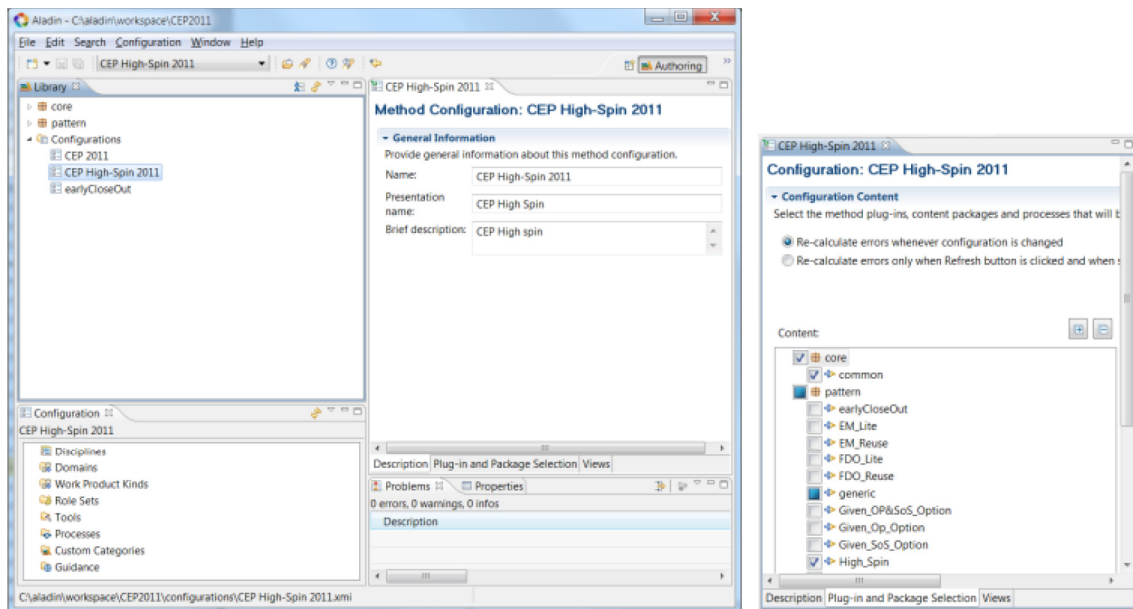


Figure 22: Method Configuration

## C.3 Modification of method elements

### C.3.1 Roles

A new role HS\_Client\_representative is created within an HS\_Role package.

An HS\_CET\_Lead\_Assistant\_role is also created, which will contribute variability to the original role, as the Sprint\_Notebook (which will be created later) will be under their responsibility (see Figure 23).

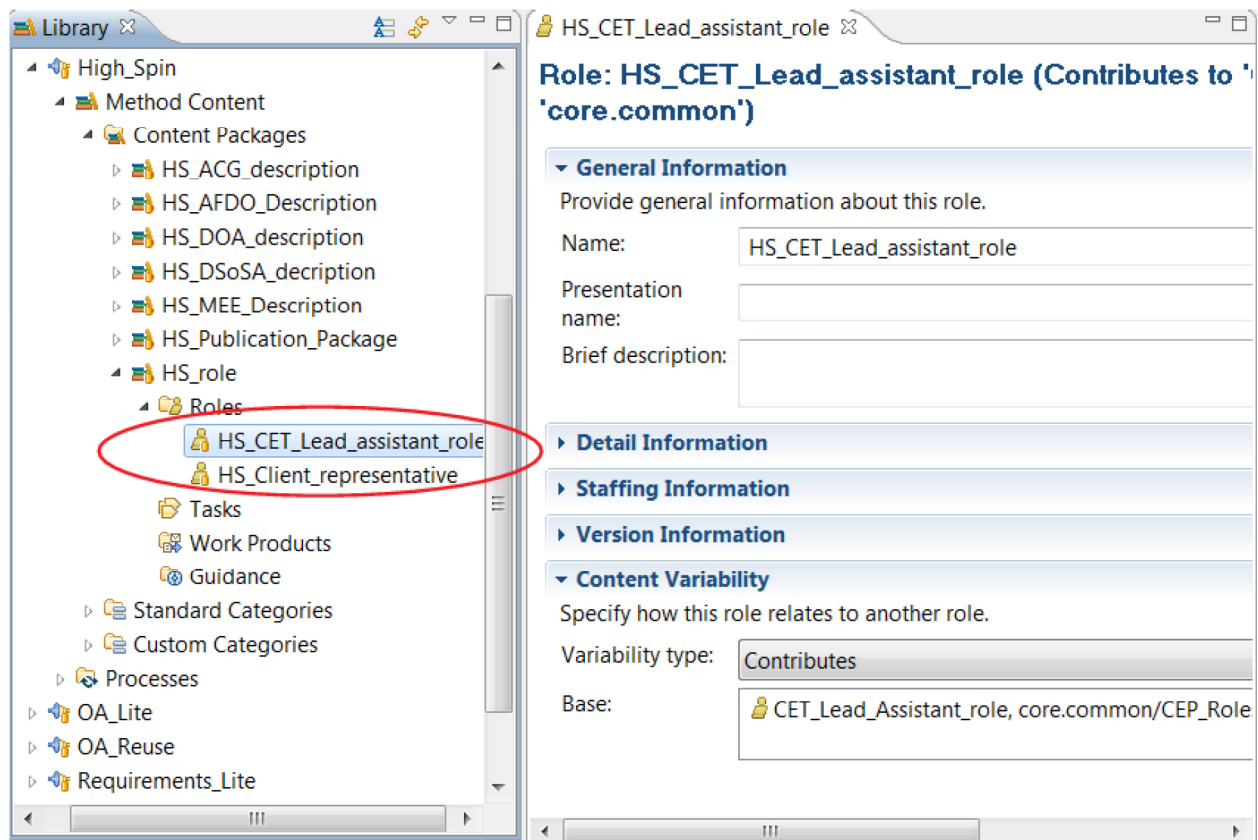


Figure 23: CET Lead Assistant Role Creation

An HS\_Stakeholders role set is created under the Role sets standard category, which contributes to the Stakeholders role set to include the client representative role.

### C.3.2 Artifacts

The new artifact Sprint Notebook is used for lightweight planning. It eliminates the need to use Progress\_report\_deliverable, Current\_period\_progress\_artifact and Outstanding\_issues\_artifact artifacts. This new artifact is created within the HS\_MEE package, as illustrated in Figure 24.

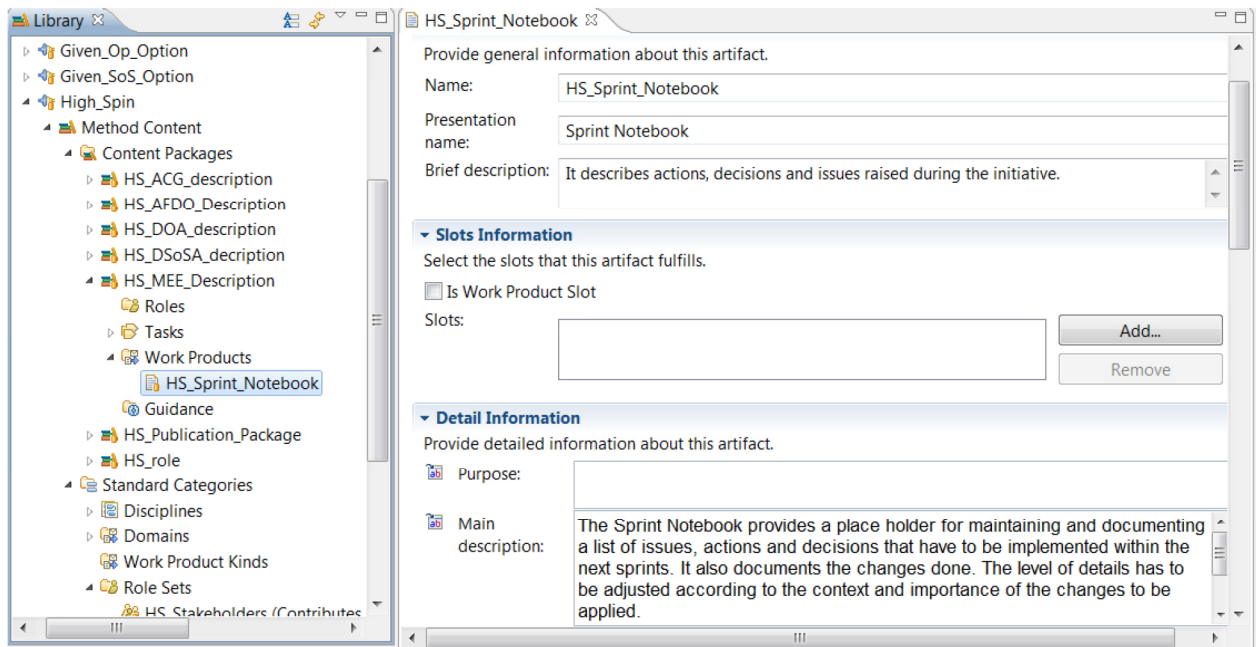


Figure 24: Sprint Notebook Artifact Creation

A HS\_Management\_domain is created under the Domains standard category. It contributes to the original Management domain within the core.common/standard categories. This domain will include the newly created HS\_Sprint notebook.

### C.3.3 Tasks

*There are a lot of changes to implement here: new tasks have to be created, while others have to be modified to take into account the new Client representative role. Changes are listed by package. Figure 25: Task Modification Using the Contributes Variability Type*

shows a typical modification to an existing task using the Contributes variability type.



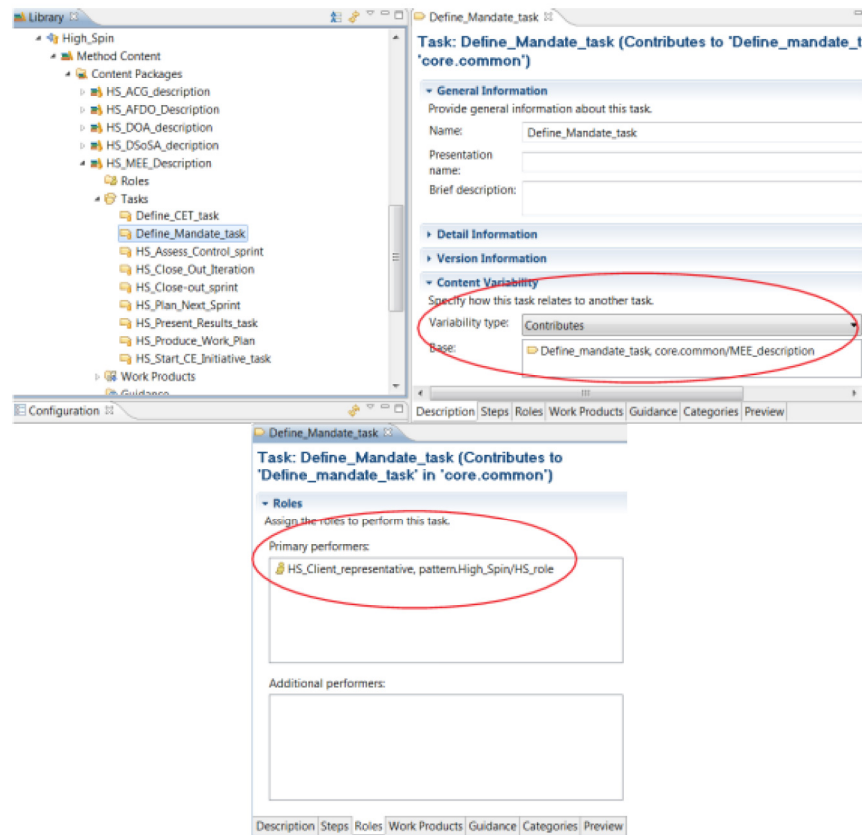


Figure 25: Task Modification Using the Contributes Variability Type

### C.3.4 Manage engineering effort package (HS\_MEE)

The following tables list the new and modified tasks of the MEE package.

Table 13: New tasks – MEE package

Name	Justifications
Assess_and_Control_Sprint_task	<p>The original Assess and Control task is kept and used in the Inception and Completion stages.</p> <p>Use of Sprint Notebook instead of progress_report_deliverable, current_period_progress_artifact, outstanding_issues_artifact</p> <p>Steps changed.</p>
Close-out_Sprint_task	<p>Need to keep close-out to end Inception and completion.</p> <p>Similar to close-out but specific to a sprint.</p>

Table 14: Modifications to Existing Tasks – MEE Package

<b>Name (variability)</b>	<b>Field</b>	<b>Modifications</b>
Define_CET_task (Extends and Replaces)	Primary performers	Remove Capability Manager Add Client Representative
Define_Mandate_task Close-Out_task Present_Results_artifact Prune_SoS_Option_task Prune_Operational_Options_task Identify_Stakeholders_task (Contributes)	Primary performers	Add Client Representative
Start_CE_Initiative_task (Contributes)	Additional performers	Add Client Representative
Plan_Next_Iteration_task (Extends and Replaces)	Name	Replace by “Plan Next Sprint”
	Brief description	Replace by “This task develops work plan, deliverables schedule and budget plans for next sprint.”
	Purpose	Replace by “To complete the work plan for next sprint.”
	Steps	Resulting steps: <ul style="list-style-type: none"> <li>• Update work plan for next sprint</li> <li>• Update work products schedule for next sprint</li> <li>• Update budget for next sprint</li> </ul>
	Additional performers	Add Client Representative
	Mandatory inputs	Replace Sprint_Notebook_artifact Work_Plan_artifact
	Outputs	Replace Sprint_Notebook_artifact Work_Plan_artifact
Produce Work Plan (Extends and Replaces)	Brief description	Replace by “This task develops CE initiative work plan, as well as its sprints, schedule and budget.”
	Steps	Replace by <ul style="list-style-type: none"> <li>• Develop the global and sprint breakdown structure</li> </ul>

Name (variability)	Field	Modifications
		<ul style="list-style-type: none"> <li>• Generate global and sprint schedule</li> <li>• Define deliverables schedules</li> <li>• Budget for each sprint</li> </ul>

### C.3.5 Analyze capability gap (HS\_ACG)

The following table lists task modifications needed for the ACG package.

*Table 15: Modified Tasks – ACG package*

Name (variability)	Field	Modifications
Analyze_Strategic_Factors_task Define_High-Level_Needs_and_Constraints_task Establish_Requirement_Priority_task Identify_Strategic_Decision_Criteria_task  (Contributes)	Additional performers	Add Client Representative

### C.3.6 Define operational activity (HS\_DOA)

The following table lists task modifications needed for the DOA package.

*Table 16: Modified tasks – DOA package*

Name (variability)	Field	Modifications
Analyze_Operational_Activity_Behaviour_task Analyze_Operational_Scenarios_task Develop_CEDF_Performance_Characteristics_task Develop_Overarching_Operational_Concept_task Identify_Operational_Activity_Interfaces_task Identify_Organizational_Composition_task Identify_Risks_task Outline_Operational_Options_task Perform_Operational_Activity_Decomposition_task Refine_CEDF_Performance_Characteristics_task	Additional performers	Add Client Representative

Name (variability)	Field	Modifications
Refine_Overarching_Operational_Concept_task (Contributes)		

### C.3.7 Analyze force development options (HS\_AFDO)

The following table lists task modifications needed for the AFDO package

*Table 17: Modified tasks – AFDO package*

Name (variability)	Field	Modifications
Develop_CEDF_Model_task Optimize_FDO_task Review_Operational_Options_task Review_Requirements_task Review_Sos_Requirements_task Review_SoS_Options_task (Contributes)	Additional performers	Add Client Representative

### C.3.8 Develop SoS options (HS\_DSOSA)

The following table lists task modifications needed for the DSOSA package.

*Table 18: Modified tasks – DSOSA package*

Name (variability)	Field	Modifications
Analyze_Existing_Systems_task Analyze_Potential_System_task Prune_SoS_Option_task (Contributes)	Additional performers	Add Client Representative

## C.4 Custom categories

### C.4.1 Role Set, domain and discipline

As mentioned above, the Stakeholder role set, the Management domain and the Manage Engineering Effort discipline are modified to take into account the new, modified or removed role, artifact and tasks that belong to them.

### C.4.2 Deliverables

The Progress Report Review deliverable is deleted from the deliverables custom category, as it is replaced by the Sprint\_notebook artifact.

### C.4.3 Copyrights and navigation views

Within the HS\_Publication package, the About page is modified through the Extends and Replaces variability. Its new description reads “Capability Engineering Process 2011 – High-Spin Pattern. Last update September 2012.”

The Copyright page is also adjusted through the same variability mechanism to change the description field to: “Produced by DRDC Valcartier • Current version CEP 2011 – High-spin Pattern • Last update September 2012. © Her Majesty the Queen as represented by the Minister of National Defence, 2012.”

Within the custom category, the navigation view is modified through the Extends and Replaces variability, in order to change the navigation menu name (CEP High-Spin 2011).

## C.5 Modification of process elements

The changes within the process packages (see Figure 26) are:

- Creation of a new stage (HS\_Sprint).
- Creation of the Manage\_sprint and Organize\_next\_Sprint activities.
- Modification to the delivery process.

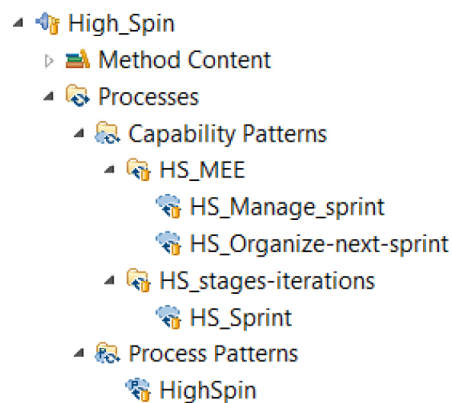


Figure 26: Process Structure Modification

### C.5.1 New activities

The HS\_MEE package is created. It contains two new capability patterns: HS\_Manage\_Sprint and HS\_Organize\_next\_sprint, which replace the original ones. Their relevant tasks are chosen and their workflow is adjusted through the activity graph feature.

### C.5.2 New stage

The HS\_Sprint capability pattern is created under the HS\_stage-iterations process package. Its workflow is built with the activities from the comprehension and elaboration stage (from the pattern.generic/stages-iteration package) and the two newly created activities (within the pattern.HighSpin/HS-stages-iteration package).

Activities are added to the workflow by using the “Apply pattern/Extend” command and choosing the appropriate activity. The activity diagram is also rebuilt. The workflow and task dependencies are automatically adjusted by the arrows between tasks within the diagram (see Figure 27).

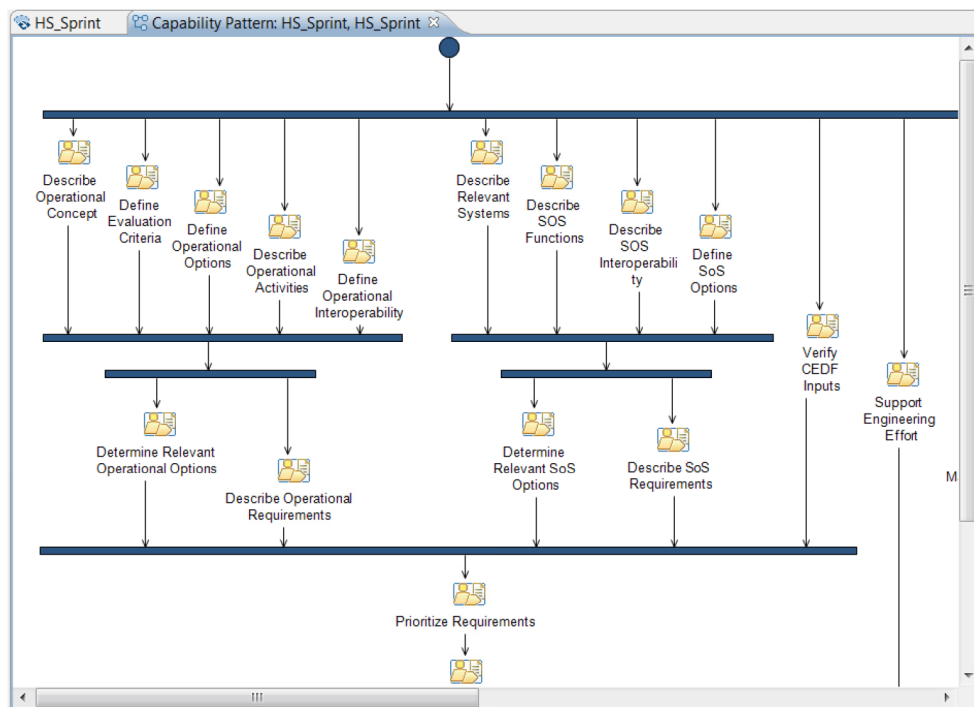


Figure 27: Activity diagram of the HS\_Sprint stage

### C.5.3 Delivery process

The delivery process is made within the High-Spin Process Pattern workflow tab by applying the Inception, HS\_Sprint and Completion capability pattern, from the pattern.generic and pattern.highspin plug-ins.



## C.6 Publication

### C.6.1 Configuration plug-in and package selection

Adjustments are made to the process pattern configuration by selecting the core.common as well as the Generic and High-Spin plug-ins. It is important to subtract the custom category views from the CEP Generic plug-in in order to avoid multiple redirections at publication time. Multiple redirections of content result in undesired behaviour within the published site (see Figure 28).

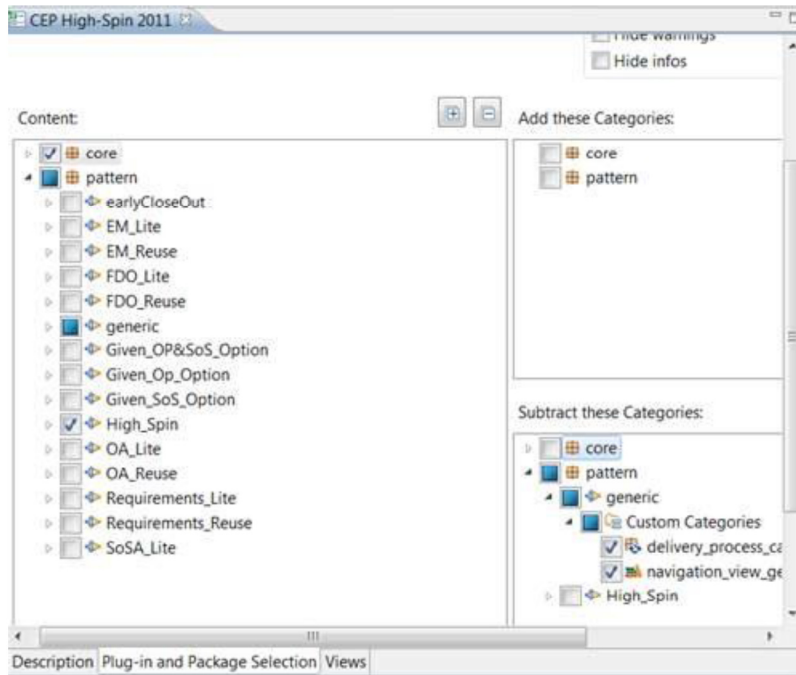


Figure 28: Package Selection

## List of symbols/abbreviations/acronyms/initialisms

---

ACG	Analyze Capability Gap
AFDO	Assess Force Development Option
ARP	Applied Research Project
CapDEM	Capability Definition Engineering and Management
C-AS4M	Counter Anti-Ship Supersonic Sea Skimming Missile
CEP	Capability Engineering Process
CET	Capability Engineering Team
DND	Department of National Defence
DOA	Develop Operational Option
DRDC	Defence Research and Development Canada
DRDKIM	Director Research and Development Knowledge and Information Management
DSOSA	Develop System of System Option
EPF	Eclipse Process Framework
GBAMD	Ground-Based Air and Munitions Defence
HS	High-Spin
ISO/IEC	International Standard Organization/International Electrotechnic Commission
ITIL	Information Technology Infrastructure Library
JFS	Joint Fire Support
MAM	Method Authoring Method
MEE	Manage Engineering Effort
OMG	Object Management Group
OpenUP	Open Unified Process
PACEM-RE	Patterns and Agility for Capability Engineering Methodology – Requirements Engineering
PMBok	Project Management Body of Knowledge
R&D	Research and Development
RTE	Rich Text Editor
SPEM	Software Process Engineering Metamodel
TDP	Technical Demonstration Program
TTCP-TP4	The Technical Cooperation Program – Technical Panel 4

UMA	Unified Method Authoring
UMF	Unified Method Framework
UML	Unified Modeling Language

# Glossary

---

## **Activity**

The main building block for processes. Activities are made of tasks or activities organized in a workflow.

## **Aladin**

The software tool, based on the Eclipse Process Framework, is used to document and publish CEP process patterns according to the CEP Documentation Framework.

## **Artifact**

An artifact is a work product in the input or output of a task.

## **CEP Documentation Framework**

The set of rules, recommended structures, formats and attributes that are used to document capability engineering process patterns.

## **Deliverable**

A deliverable is a collection of artifacts. Deliverables are the final results of a delivery process.

## **Delivery Process**

The process covering the lifecycle of the project, from start to end.

## **Discipline**

A discipline is made from related tasks.

## **Method**

The way of doing something. It implies generally a logical sequence as well as planning and a mix of knowledge and behaviour.

## **Method Generation**

The transformation of process patterns toward a generic process or method by selection and assembly of process method chunks.

## **Methodology**

The scientific study of methods. Methodologies are families of methods applying similar techniques.

## **Process**

A repeatable documentation of steps, templates and roles description that are needed in order to reach a goal.

**Process Enactment**

The action of applying a process to a particular project. The instantiation of a process.

**Process Instance**

The application of a process to a specific project, made by specific personnel with specific tools and deliverables.

**Process Pattern**

A process model proposing a generic solution to different challenges encountered in real-life process execution. Such patterns can be applied to specific situations and are used to generate tailored processes.

**Process Pattern Generation**

The creation of process patterns within a documentation framework.

**Role**

A role defines a profile of process actors. It describes skills, responsibilities and competencies. A role performs tasks.

**Role set**

A Role set is a grouping of roles.

**Stage**

A stage is a significant activity within a delivery process, leading to a milestone. Stages are the first level of activity decomposition within a delivery process.

**Step**

The sub-division of work to be done within a task.

**Task**

A unit of work which can be assigned to one or more roles and may produce artifacts. The base element of an activity.

This page intentionally left blank



DOCUMENT CONTROL DATA		
(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)		
1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.)  Defence R&D Canada – Valcartier 2459 Pie-XI Blvd North Quebec (Quebec) G3J 1X5 Canada	2. SECURITY CLASSIFICATION (Overall security classification of the document including special warning terms if applicable.)  UNCLASSIFIED (NON-CONTROLLED GOODS) DMC A REVIEW: GCEC June 2010	
3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)  Capability Engineering Process Documentation Framework: How to document a CEP within Aladin		
4. AUTHORS (last name, followed by initials – ranks, titles, etc. not to be used)  Nécaille, C.; Lalancette, C.		
5. DATE OF PUBLICATION (Month and year of publication of document.)  February 2013	6a. NO. OF PAGES (Total containing information, including Annexes, Appendices, etc.)  94	6b. NO. OF REFS (Total cited in document.)  25
7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)  Technical Memorandum		
8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)  Defence R&D Canada – Valcartier 2459 Pie-XI Blvd North Quebec (Quebec) G3J 1X5 Canada		
9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)  10BD	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)	
10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)  DRDC Valcartier TM 2013-176	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)	
11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)  Unlimited		
12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected.)  Unlimited		

13. **ABSTRACT** (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

The PACEM-RE Project (Patterns and Agility for Capability Engineering Methodology-Requirement Engineering) needed to formalize how documenting engineering processes were documented to be able to insert the concept of process patterns. This report details the Capability Engineering Process Documentation Framework established during the PACEM-RE Project. It introduces various background elements and presents the components and structure of the process documentation through a process engineering repository.

Le projet PACEM-RE (Pattern et agilité pour les méthodes d'ingénierie de capacité) nécessitait l'établissement d'un cadre plus formel de documentation pour la publication des processus d'ingénierie, afin de pouvoir utiliser le concept de *process pattern*. Ce rapport détaille le cadre de documentation des processus d'ingénierie de capacité établi durant ce projet. Il introduit différentes notions de bases et présente les composantes et la structure du cadre de documentation par le biais d'un entrepôt de processus d'ingénierie.

14. **KEYWORDS, DESCRIPTORS or IDENTIFIERS** (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Capability Engineering Process; CEP; Documentation Framework



## **Defence R&D Canada**

Canada's Leader in Defence  
and National Security  
Science and Technology

## **R & D pour la défense Canada**

Chef de file au Canada en matière  
de science et de technologie pour  
la défense et la sécurité nationale



**[www.drdc-rddc.gc.ca](http://www.drdc-rddc.gc.ca)**