

POLITECNICO DI TORINO

Department of Mechanical and Aerospace Engineering



Bachelor's Degree in Aerospace Engineering

Final Project

Feasibility Study for a Knowledge Base of Enterprise Practices

Academic Supervisor:

Prof. Paolo Maggiore

Company Supervisor:

Engr. Francesco Lanteri

Candidate:

Filippo Montanari

September 2018

TABLE OF CONTENT

<i>Preface</i>	v
<i>Chapter 1: Feasibility Study: Goals, Development Approach and Solutions</i>	
1.1 Goals of the study	1
1.2 Presentation of the product	1
1.3 Method library development	1
1.4 Developer level goals and derived solutions	2
<i>Chapter 2: TSO Practices Library</i>	
2.1 Introduction	5
2.2 TPL web-site	5
2.3 Library internal architecture	10
2.4 TPL modules	11
2.5 Hierarchy of references	14
2.6 Level of specification of TPL content	15
2.7 Exportability to project management software	16
<i>Chapter 3: Study Conclusions: Technical and Economic Considerations</i>	
3.1 Collected evidence and lessons learned	17
3.2 Evaluation of possible benefits	17
3.3 Estimation of costs	18
3.4 Summary points	19
<i>Appendix A: Introduction to Systems Engineering</i>	
A.1 Introduction	21
A.2 Definitions	21
A.3 Development phasing	22
A.4 The systems engineering process	23
A.5 Life cycle integration	24
A.6 Freedom degrees of systems engineering management	24
A.7 The process in detail	25
A.8 Summary points	27
<i>Appendix B: Introduction to the CMMI® Institute Development Model</i>	
B.1 Introduction	29
B.2 Capability maturity models	29
B.3 Process area components	29
B.4 Capability and maturity levels	31
B.5 Process areas	33
B.6 Process improvement program	34
B.7 CMMI models	34

B.8	Process institutionalization	35
B.9	Organizational process assets	36

Appendix C: Introduction to EPF Composer

C.1	About the EPF Project	37
C.2	Exemplary tool: EPF Composer	37
C.3	Plug-ins logics	38
C.4	EPF method framework	39
C.5	EPF Composer elements	40

<i>References and Resources List</i>	43
--	----

<i>Acronyms</i>	45
-----------------------	----

TABLE OF FIGURES

<i>Figure 1.1 – Project development approach</i>	2
<i>Figure 1.2 – Dependencies between user-level goals, developer-level goals and developer-level solutions</i>	3
<i>Figure 2.1 – TSO Practices Library web-site Home Page</i>	6
<i>Figure 2.2 – ‘Peer Review’ practice web-page</i>	7
<i>Figure 2.3 – ‘MBSE process: GBTS SW Development and Maintenance’ delivery process web-page</i>	8
<i>Figure 2.4 – ‘Process Tailoring – method A’ activity diagram</i>	9
<i>Figure 2.5 – TPL modules and content division</i>	10
<i>Figure 2.6 – ‘coreTSO’ module content</i>	11
<i>Figure 2.7 – ‘processTSO’ module content</i>	12
<i>Figure 2.8 – ‘practiceTSO’ module content</i>	13
<i>Figure 2.9 – ‘publishTSO’ module content</i>	14
<i>Figure 2.10 – Main ‘publish’ plug-in content</i>	14
<i>Figure 2.11 – Scheme of an allowed tree of references</i>	15
<i>Figure 2.12 – Case study: ECS design process exported to Microsoft Project</i>	16
<i>Figure 3.1 – Role of collected evidence and lessons learned in TPL development cycle</i>	17
<i>Figure 3.2 – Case Study of instantiation</i>	18
<i>Figure A.1 – The three activities of systems engineering management</i>	22
<i>Figure A.2 – The systems engineering process</i>	23
<i>Figure A.3 – Detailed systems engineering process</i>	25
<i>Figure A.4 – Systems engineering and verification</i>	26
<i>Figure B.1 – CMMI model components</i>	30
<i>Figure B.2 – Process areas in the continuous and staged representations</i>	32
<i>Figure B.3 – Table of process areas, categories, and maturity levels</i>	35
<i>Figure C.1 – Exemplary tool: EPF Composer</i>	37
<i>Figure C.2 – Setting of a new method plug-in</i>	39
<i>Figure C.3 – EPF method framework</i>	39



**POLITECNICO
DI TORINO**



Preface

The bachelor's degree final project was developed in cooperation with Leonardo Aircraft Division as an internship project together with the colleague Pippa Martina.

This project – proposed by the Training Software Organization of LAD – aims at testing and evaluating new methods and tools reflecting the current state-of-the-art in the process engineering context.

Systems engineering is nowadays one of the fastest growing engineering fields and the precious support that process engineering brings is undeniable. As a result, keeping up with the emerging technologies is fundamental in order to maintain a durable and safe company business.

In particular, for what concerns process engineering, two main problems have to be faced when deploying a new product development process. Firstly, the working team needs to be instructed about the different roles and responsibilities to cover within the process, and secondly, but not for importance, a clear and unambiguous view on the activity sequence should be provided.

Currently, the Training Systems Organization of LAD can count on its dedicated Organizational Process Asset Library (OPAL) to develop its products in accordance with the systems engineering principles and the Capability Maturity Model Integration (CMMI) approach. The OPAL is made up of static documents, divided by process areas as required by CMMI, but like any static knowledge base, it does not really permit an easy, quick and focused consultation of its content.

It was within this context that the internship project was carried out; the final purpose, as the name may suggest, was to study the implementation of a knowledge base of enterprise practices in its technical and economic aspects.

Moreover, two main characteristics were required to the final product: to feature a user-friendly approach, so as to permit a quick and focused consultation of the desired pieces of content, and to present a process tailoring section explaining to users how to instantiate a company standard process on a customer's specific project.

The knowledge base prototype was the final tangible product of the study, but the most important outcome is the amount of 'lessons learned' and concepts extrapolated during the prototyping activities.

The implementation of the knowledge base was carried out with the framework tool EPF Composer, developed within the EPF project of the Eclipse Foundation, in cooperation with IBM.

The Leonardo Company Tutor that personally followed this internship project is the Engineer Francesco Lanteri, of the Integrated Training Systems Organization of LAD.

Chapter 1

FEASIBILITY STUDY: GOALS, DEVELOPMENT APPROACH AND SOLUTIONS

1.1 GOALS OF THE STUDY

This study, as stated in the *Preface*, was carried out in cooperation with the Training Systems Organization of Leonardo Aircraft Division.

All the projects and processes for product development of the Training Systems Organization are currently regulated by a LAD standard approach, based on official documents compliant with the CMMI level 3 mandatory principles and guidelines.

With this study, it was explored the possibility to enhance the current approach, improving its user-friendly aspects and giving to customers the chance to instantiate a standard process on a specific project.

In summary, the study tested the possibility of providing LAD TSO employees with:

- a knowledge base of the LAD TSO official standard processes and practices featuring an easy and user-friendly consultation;
- a set of tailoring guidelines and procedures useful to instantiate a standard process on a specific project, always respecting the official LAD TSO acceptance criteria for product development processes.

Beside the technical aspects, a first evaluation of the economic impact was conducted.

1.2 PRESENTATION OF THE PRODUCT

Before proceeding further, the product of the study is hereafter briefly introduced.

It consists of a method library prototype edited with the framework tool EPF Composer, with the final purpose of publishing an html web-page of organized content.

EPF Composer is – as stated by the Eclipse Foundation – an open-source tool platform designed for process engineers and project manager to author, tailor and publish methods and processes for development organizations and projects. Read ‘*Appendix C: Introduction to EPF Composer*’ to get a basic knowledge of EPF Composer and its fundamentals.

TSO Practices Library (TPL) is the name given to the developed knowledge base. It was filled with content extrapolated from the Organization’ Process Asset Library (OPAL) of Training Systems Organization, that contains the official enterprise processes for software products developments, divided by process areas. See paragraph ‘*B.9 Organizational Process Assets*’ for further information.

The second capability given to TPL is, as previously stated, to provide the user with a library of reusable content, as well as an amount of procedures explaining how to tailor and build a defined process – fitting with a customer’s specific project – starting from the existing standard processes and process elements.

Beside OPAL consultation and tailoring features, TPL also contains guidelines and procedures useful to navigate inside TPL web-site and expanding the method library by adding and integrating new plug-ins.

The TSO Practices Library architecture and content is discussed in detail in ‘*Chapter 2: TSO Practices Library*’.

1.3 METHOD LIBRARY DEVELOPMENT

The development of TSO Practices method library required the preliminary definition of a development cycle featuring an explorative approach, firstly necessary to define the user level goals.

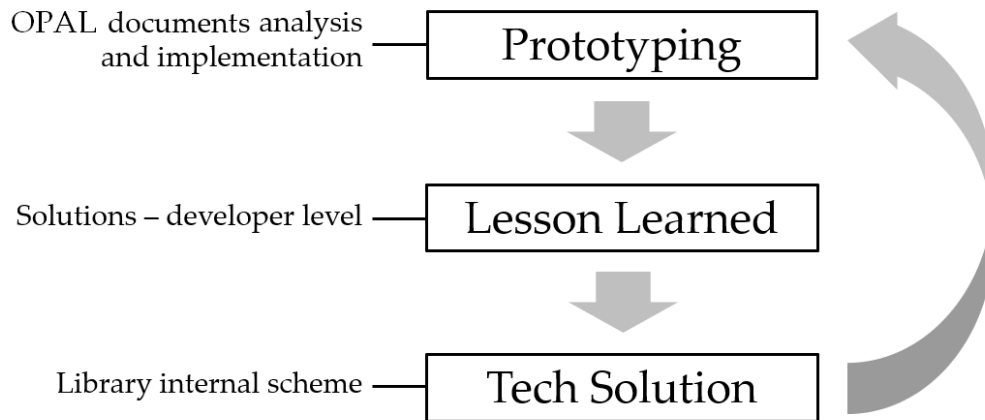


Figure 1.1 – Project development approach

As illustrated in *Figure 1.1*, the adopted approach presents three main phases:

- *a prototyping phase*, consisting in the analysis and the implementation of OPAL official documentation into the TSO Practices Library prototype;
- *a learning phase*, in which the prototyping evidences are turned into formally explained ‘lesson learned’. In this case, the OPAL analysis and implementation provides, besides mere inputs for technical adjustments, information regarding the developer level goals, explained in detail in the next chapters;
- *a technical solution definition phase*, which gives information about the practical development of the method library, especially for what concerns about the possible adjustments to the current library internal architecture. This topic is discussed in detail in the next chapter; also see ‘*Appendix C: Introduction to EPC Composer*’ for further information about EPF concepts regarding method library architecture.

These three steps were iterative and progressive, since every trial did not lead to the best solution but was a further step toward a suitable one.

1.4 DEVELOPER LEVEL GOALS AND DERIVED SOLUTIONS

In this paragraph are set out the developer level goals and – not less important – the developer level solutions, as well as the interrelationships

between them. *Figure 1.2* shows in general terms the dependencies between user-level goals, developer-level goals and developer-level solutions.

Respecting the principles hereafter explained makes possible to define a suitable method library internal architecture, explained in detail in ‘*Chapter 2: TSO Practices Library*’.

Developer level goals were written considering the final and user-level purpose of the study, as well as the potentiality and the limits of the framework tool. Here they are listed and explained:

Content Reuse

This expression refers to the chance for TPL users to have access to a wide repository of standard content elements, with the opportunity to call up, withdraw and reuse them in order to avoid creating new content (saving time and space on disk) and to tailor a standard process so as to define a specific one.

Pursuing this goal brings the following advantages:

- users benefit with a time saving, since there is no need to create all the content from scratch,
- tailoring activities are enhanced for what concerns rapidity and disambiguation,
- library maintainability and navigability are increased,
- library files are lighter, and the tool best manages with them.

Easy Maintenance

Maintenance activities are those actions aimed to modify, expand or enhance TPL content. For many of these activities, dedicated sections of

TPL web-guide were created, in order to contain guidelines and procedures concerning maintenance activities.

Keeping an easy level of maintenance and properly maintaining TPL brings the following advantages:

- maintenance activities are encouraged,
- finding a certain piece of content inside the method library is much easier and users' time is saved,
- the occurrence of problems related to addressing and plug-ins references is reduced.

Multiple Entry Points

An entry point is a virtual view on library content with certain characteristics and organizational features.

TPL aims to provide the users with the possibility to choose the entry point that best suits their consultation needs (what they exactly need to know), and that must be possible at a rather high level. For the specific case of TPL, entry points are divided in 'practice' entry points and 'process' entry points, that again are respectively sub-divided depending on process areas, product types and development approach.

Example: a TPL user is involved in the development of an on-board avionic software with a MBSE approach. Whenever this user needs a piece of information regarding the requirements development, it may be selectively found by choosing the entry point called '*Model Based Engineering Practices: MBSE - Requirements Development*'. The name is self-explanatory; this entry point brings to the web-page of the LAD TSO

requirements development practice suitable to carry on a software development process featuring the MBSE approach.

Setting many different entry points presents the following advantages:

- web-site navigability is enhanced,
- a focused consultation of specific pieces of content is permitted.

Developer level solutions are to be kept into account while developing method library technical arrangements. Namely they are:

Content Standardization

With standardization is meant any action aimed to align all the content and process elements belonging to a same type.

This topic also involves the evaluation of the correct specification level for every different TPL piece of content, a topic not to neglect considering the importance that 'reuse' has in the whole project. See paragraph '*2.6 Level of Specification of TPL Content*' for further information.

Standardize content permits to reach the goals:

- *Content Reuse:*
content is much more suitable for reuse, since content elements and specification levels are aligned and reflect user's expectations.
- *Easy Maintenance:*
employees in charge of TPL maintenance may count on the fact that content general features and addressing are standardized and recorded.

Modularity

To be compliant with this solution, TPL architecture features different kinds of plug-ins in which content is logically divided, as well as a way of grouping them which depends on the project to

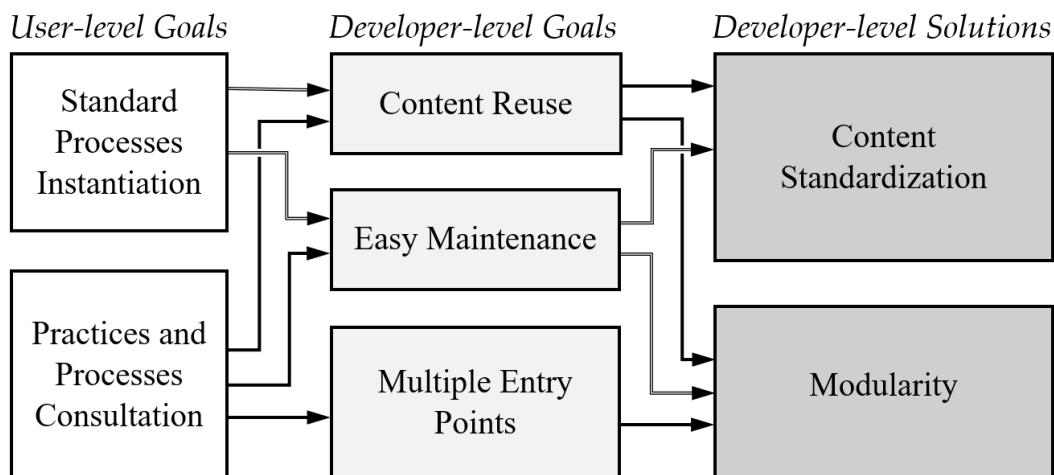


Figure 1.2 – Dependencies between user-level goals, developer-level goals and developer-level solutions

which they are dedicated.

Concerning the first part, the different names based on the different content kinds are discussed in ‘*Chapter 2: TSO Practices Library*’.

The second point, instead, refers to the fact that the content belonging exclusively to a customer’s specific project may be wholly found inside a designated set of exportable plug-ins.

Finally, granting TPL expansibility is also a fundamental aspect of modularity.

Modularity contributes to reach the following goals:

- *Content Reuse:*
adopting a modular architecture permits a clear and defined chain of references, indispensable to recall and reuse content of other plug-ins (of a different kind and/or belonging to a different set).
- *Easy Maintenance:*
employees in charge of maintaining TPL may count on the fact that content related to a circumscribed topic is located inside the plug-ins designated for that topic or, at most, in a referenced plug-in. Moreover, modularity permits to import and export plug-ins.
- *Multiple Entry Points:*
entry points and plug-ins architecture (modules) find a very close way of development. E.g. the source element of the entry point ‘*MBSE - Requirements Development*’ may be found inside the same set of plug-ins (modules) that also contains the whole ‘*MBSE - Requirements Development*’ content elements and processes.

Chapter 2

TSO PRACTICES LIBRARY

2.1 INTRODUCTION

TSO Practices Library prototype is the final tangible product of the internship project with Leonardo Aircraft Division.

This chapter firstly introduces the html web-site of TPL, obtained by publishing the EPF source method library. Every web-site feature offered to the user is here explained and commented. After that, also the TPL source method library is shown in detail and explained, in order to permit a better understanding of what lays behind TPL web-site and what are some of the internal mechanisms of the framework tool EPF Composer.

2.2 TPL WEB-SITE

TPL web-site is composed of a multitude of html pages and hyperlinks necessary to connect them.

Besides the main project goals – that were implementing TSO OPAL documentation in a web library and providing a process tailoring systematic approach – other implicit requirements were to be considered, such as finding a suitable and user-friendly setting for the web-site so as to keep navigability at a rather easy level. Below is a list of TPL navigation pages created to permit the consultation of TPL features.

TPL Home Page

Opening the TPL web-site html file (index) brings to the TPL home page.

As shown by *Figure 2.1*, in this page there are six fields introducing different TPL features:

- a ‘Welcome’ field, explaining project general purposes (user-level goals),
- a field to explain developer-level goals and solutions,
- an ‘Help’ field, to solve user’s doubts about TPL navigation and library development, and that reports a hyperlink to TPL Guide Home Page,
- a list of the external resources,
- a process tailoring brief description and a hyperlink to process tailoring procedure,

- a field introducing practices and standard processes consultation and, lastly,
- a field designated for TPL expansion, that again brings to TPL Guide Home Page.

TPL Guide Home Page

This navigation page is reached by clicking the hyperlinks in TPL home page or, alternatively, in the web-site sidebar.

It contains three fields concerning different features:

- a ‘Navigation Guidelines’ field, introducing the concept of web-site navigability and presenting hyperlinks leading to these guidelines,
- a field explaining the TPL development approach and a hyperlink to the modelled process concerning it,
- a field introducing TPL expansions and the hyperlink leading to the process explaining how to create them.

Practices Navigation Web-page

‘Practices’ hyperlink in TPL Home leads to the main navigation page for TPL practices, that sorts them by process areas category and development approach: Project Management Practices, Process Management Practices, Company Standard Engineering Practices, Model Based Engineering Practices, and Support Practices.

The adopted practices classification was developed by the CMMI Institute and gathers process areas – to which practices are related – into four categories depending on their technical, economical or administrative aspect. See ‘*Appendix B: Introduction to the CMMI® Institute Development Model*’ for further information about the CMMI model and process areas.

Processes Navigation Web-page

Reached by clicking the ‘Processes’ hyperlink of TPL Home, processes navigation main page presents four hyperlinks leading to four different delivery processes, each of them having its own combination of product type (Aircraft or GBTS software) and development approach (Company Standard or MBSE), namely they are:

Welcome to LAD TSO Practices Library Website

TSO Practices Library Website aims to support process engineers in their project management activities by:

1. Providing them with a Knowledge Base of standard processes and practices;
2. Giving them the opportunity to tailor a Defined Process starting from the most suitable standard one.

Goals and Solutions

To fulfill the user level Goals, three are the developer level Goals of TSO Practices Library:

- Content Reuse
- Easy Maintenance
- Multiple Entry Points

and two are the development Solutions adopted:

- Content Standardization
- Modularity

Help

Need help with TPL navigation?

See [TPL Guide Website](#) for much information about how to navigate inside TPL Website, as well as an explanation concerning how TPL was developed.

Resources

- Eclipse Process Framework Project
- CMMI Institute
 - CMMI-DEV v1.3

Tailor your Process

Process Tailoring is the main point of LAD TSO Practices Library.

TPL offers many guidelines and best practices useful to tailor a Defined Process, developing it within the principles of TPL Solutions, so as to never neglect TPL Goals.

[Build up your Defined Process](#)

Consult LAD TSO Practices and Standard Processes

[Practices](#) [Processes](#)

TPL Website offers two different main entry points on LAD Systems Engineering project procedures.

The first entry point leads to Practices web-page, in which the user may select the required process area and read everything that is necessary to perform the described activities. The second entry point, instead, brings the user to the web-page containing Standard Processes, both for Aircraft and GBTS software development, each described with both Company Standard and Model Based development approach.

TPL Expansions

TSO Practices Library - source of this web-site - may be expanded and customized by any user by following the TPL Expansion procedures, described within [TPL Guide Website](#).

Figure 2.1 – TSO Practices Library web-site Home Page

- Company standard process: Aircraft Software Development and Maintenance,
- Company standard process: GBTS Software Development and Maintenance,
- MBSE process: Aircraft Software Development and Maintenance,
- MBSE process: GBTS Software Development and Maintenance.

The TPL prototype developed within this project only contains four delivery processes which feature a V-model lifecycle. Clearly, it is possible to build other delivery processes characterized by other development approaches and lifecycles, such as the prototyping model or the fountain development model.

All the main TPL navigation pages are now explained.


In the following part, the substantial content of TPL is presented, i.e. the web-pages of practices, processes and the procedures for process tailoring and TPL expansion.

Practices

Figure 2.2 shows the Peer Review Practice as an example to display the typical web-page structure of TPL – and, in general terms, all the EPF Composer – practices; a first field briefly introduces the content of the page, while a more detailed description is reported below. Field in the middle is the most important of the web-page, since there are linked the main process and method content elements indispensable to effectively complete the activities that the practice describe.

Practices are the transverse entry points on TPL content that permit to the user a focused consultation of specific pieces of information.

Practice: Peer Review














A meeting between the author and the author's peers who have studied the element under review to verify that the element will work, that it meets requirements allocated to it, and that it conforms to standards.

[Expand All Sections](#) [Collapse All Sections](#)

▣ Relationships

Content References

- Work Products
 -  Peer Review Attendees
 -  Peer Review Findings
 -  Peer Review Report to record Approval Status
 -  Peer Review Schedule
- Tasks
 -  Identify Elements for Peer Review
 -  Prepare Peer Review
 -  Conduct Peer Review
 -  Analyze Results
-  Peer Review
-  Software Development Engineer
-  Software Verification Engineer

[Back to top](#)

▣ Main Description

A **Peer Review** is formally defined in industry as a meeting between the author and the author's peers who have studied the element under review to verify that the element will work, that it meets requirements allocated to it, and that it conforms to standards. The key activities for a peer review are to prepare the material for a peer review and distribute it prior to a peer review meeting when applicable; review the material and identify potential areas for action necessary to bring the element under review into conformance with the requirements and standards; hold a review meeting (or other method of review as noted below based on the type of peer review) to obtain consensus on the problems and actions; prepare the peer review report and track the actions to closure.

[Back to top](#)

Figure 2.2 – 'Peer Review' practice web-page

Processes

Figure 2.3 shows the GBTS Software Development and Maintenance delivery process with a MBSE development approach, which is one of the four 'V-modelled' complete processes of TSO Practices Library web-site.

A typical delivery process web-page presents four views with different content; a main 'Work Breakdown Structure' view, that shows process activity diagram and the whole product development procedure; a 'Description' view; a 'Team Allocation' and a 'Work Product Usage' views, that respectively give information about roles and work products inside the process.

Differently from practices, the processes entry points give to the user a complete and top-down perspective on the product development process, information that is particularly useful to understand, learn and track the whole sequence of activities.

Tailor your Process

The 'Tailor you Process' practice is reached by clicking on the dedicated hyperlink in TPL Home Page. This practice gathers the main method content and process elements involved in process instantiation activities and, besides that, two links leading to delivery processes characterized by a different tailoring approach:

- *Method A*, in which a standard delivery process is copied in a designated directory and there it is tailored till the defined process is built. This approach is used in case the defined process is mostly the same of the standard one, a not unusual occurrence considering the standardization level that characterizes systems engineering.
- *Method B*, which explains to recall existing process elements and, whenever required, to create new ones in the set of plug-ins dedicated to the specific project, in order to build up the defined process. This approach is used whenever the final customized process to ob-

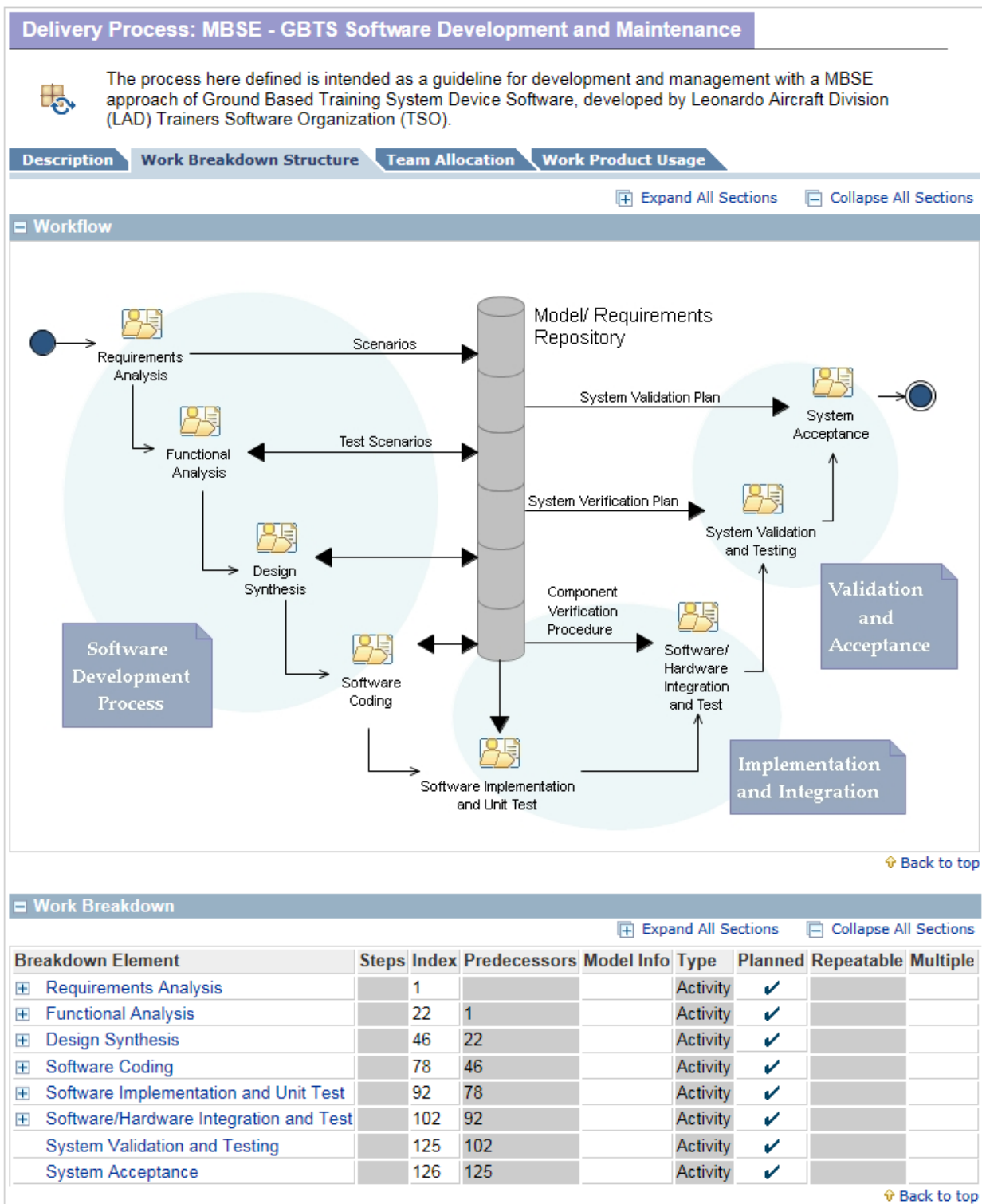


Figure 2.3 – ‘MBSE process: GBTS SW Development and Maintenance’ delivery process web-page

tain presents many differences compared to a standard process, so that it is convenient to build it from scratch using existing and new process elements.

Both these approaches were developed from the collected evidence and the lessons learned during the prototyping activities, always considering TPL

fundamental principles so as to develop effective and efficient procedures. Follows that process tailoring rigorous sequence of activities was entirely developed in this project context, since there were not external technical prompts concerning it.

Figure 2.4 illustrates Method A activity diagram; a main division of preliminary and content management activities may be noticed.

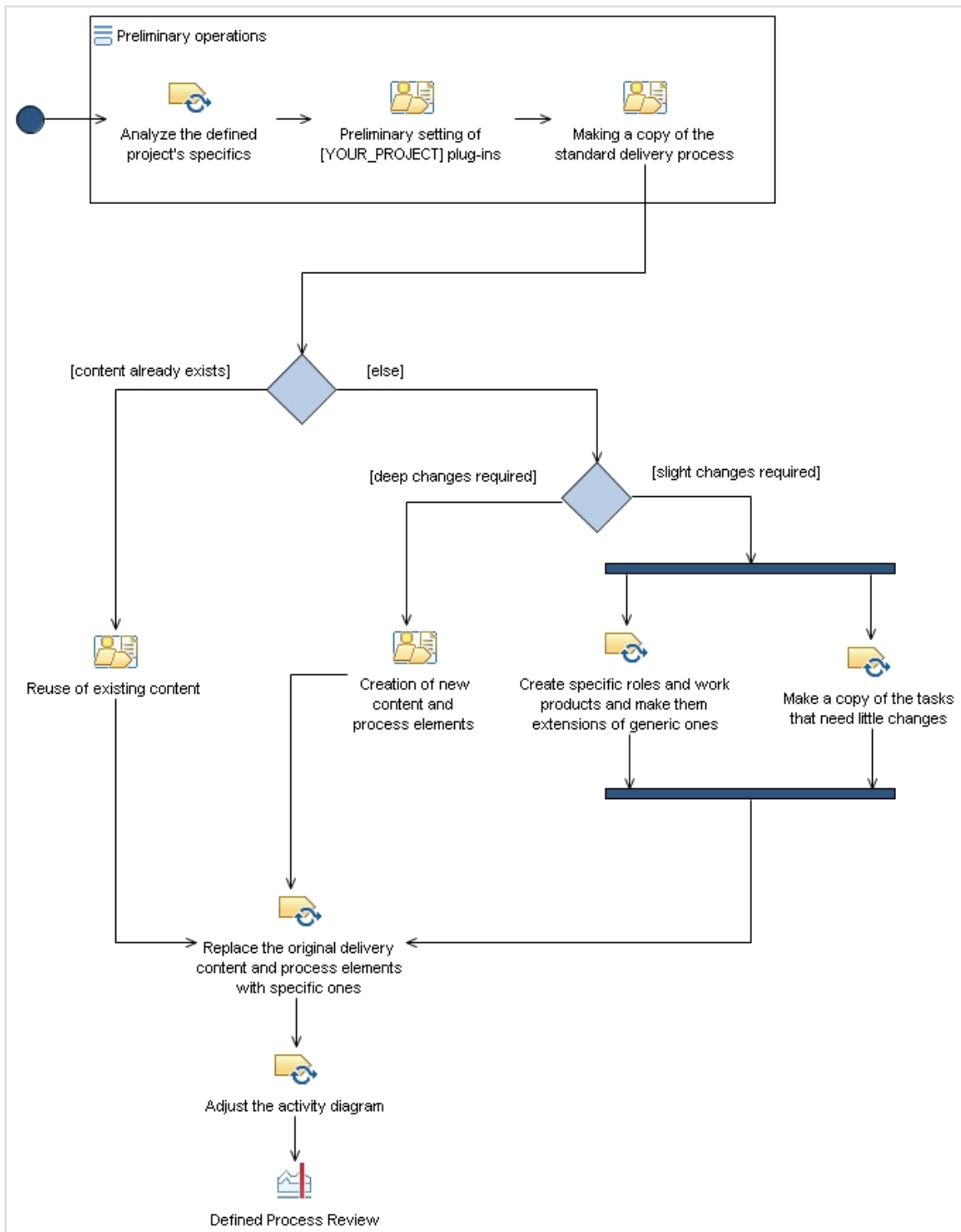


Figure 2.4 – ‘Process Tailoring – method A’ activity diagram

The first part involves analysing and eliciting the project specifics, as well as creating the necessary method library plug-ins and copying the standard delivery process, like required in Method A description.

After that, second part involves all the content management activities, i.e. recalling and copying

existing content, creating new one and replacing process and content elements where necessary.

Expand TPL

‘Expand TPL’ is part of TPL Guide web-site and explains a procedure to create a new set of plug-ins and fill them with everything necessary to be published, that is new method content (if neces-

sary), new practices, new processes and new elements required to publish and navigate. Expanding TPL is possible thanks to the fundamental modularity concept, and also it is enhanced by the possibility of reusing existing content located in other plug-ins.

TPL Development

Part of TPL Guide web-page, too, ‘TPL Development’ consists of a delivery process that reports the entire sequence of activities followed to create TPL from scratch. This process is one of the final products totally developed inside the ‘Feasibility Study’ project and, due again to modularity, many parts of it were reused in a later stage for modeling the ‘Expand TPL’ process.

Navigation Guidelines

After the two previous points, this is the last part of TPL Guide web-page. It finds its utility by considering that the published TPL web-site always has some intrinsic difficulties for what concerns navigation, reason why users could need some suggestions to effectively navigate inside it.

2.3 LIBRARY INTERNAL ARCHITECTURE

The previous paragraph aims to provide a general comprehension of what TPL web-site offers to users. This paragraph, instead, digs deeper into technical aspects, introducing TPL method library architecture and, besides that, explaining much of the framework content in detail.

Notice that this chapter deals with EPF Composer elements and functions, also using its specific terminology. See ‘Appendix C: Introduction to

EPF Composer’ (especially paragraph ‘C.5 EPF Composer Elements’) for further information about the topic.

The starting point for TPL internal architecture development was the Eclipse exemplary library ‘EPF Practices Library 1.5.1.5’. Architectures of TPL and Eclipse exemplary library present some common points, even though TPL architecture underwent a further development to be as compliant as possible with the developer-level goals and solutions defined within the project (especially ‘content reuse’ and ‘modularity’), also considering the functions and the technical limits of the tool.

As stated before, library architecture was defined and refined with an iterative process, in which a first prototyping phase was followed by a learning one and a technical-operative one. Any lesson learned by the current development cycle was used as an assimilate notion in the subsequent one. Notice, though, that the produced – and here discussed – architecture is one of the many that could comply with project goals, and it is not expected to be the best one.

In *Figure 2.1* are shown the four main kinds of TPL modules next to their typical content.

Core

The ‘core’ module contains plug-ins dedicated to most of TPL method content elements – except for tasks – that namely are roles, work products, guidance, and the main navigation pages.

Practice

In this module are gathered all the company offi-





	core	Plug-ins containing the method content elements: roles, work products and guidance.
	practice	Plug-ins containing the company official practices divided by process areas.
	process	Plug-ins containing the company official processes, process elements and tasks.
	publish	Plug-ins containing some gathering categories (such as sets of practices) and navigation pages.

Figure 2.5 – TPL modules and content division

cial practices. The aim of each of them is to provide focused information about the activities concerning a certain process area. To practices are linked the main process and method content elements regarding them.

Process

The ‘process’ module contains every element necessary to build up processes and the processes themselves. In tool terms, it contains tasks, capability patterns and delivery processes.

Publish

In the ‘publish’ module are all the custom categories used to gather similar content, e.g. a cluster of practices or a set of guidelines, with the main purpose of permitting the web-site navigation once library is published. The substantial difference between these elements and those working as navigation pages contained in ‘core’ plug-ins is that in the first case the links between elements are of the proper EPF Composer kind, while in the second case the hyperlinks consist of simple and static html content index, without the advantage of automatically changing whenever the object of the link is moved.

2.4 TPL MODULES

The four modules in which TSO Practices Library content is divided present the desinence ‘-TSO’ to distinguish them from those belonging to the original Eclipse exemplary library.

coreTSO

As shown in *Figure 2.2*, the ‘coreTSO’ module gathers many directories sorting plug-ins with different types of content, namely:

- *nav_view*, containing the framework elements of the main navigation pages, such as TPL Home Page and TPL Guide Home Page, that in practical terms are ‘guidance: supporting material’ framework elements with rich text descriptions and html hyperlinks.

- *SEP_content*, that contains the method content necessary to describe the company standard processes directly related to OPAL documents.

Module content is in turn sub-divided into dedicated directories, one for each Standard Enterprise Practice (SEP) in which OPAL documents are divided, and one for the very general content elements, such as important and recurrent roles.

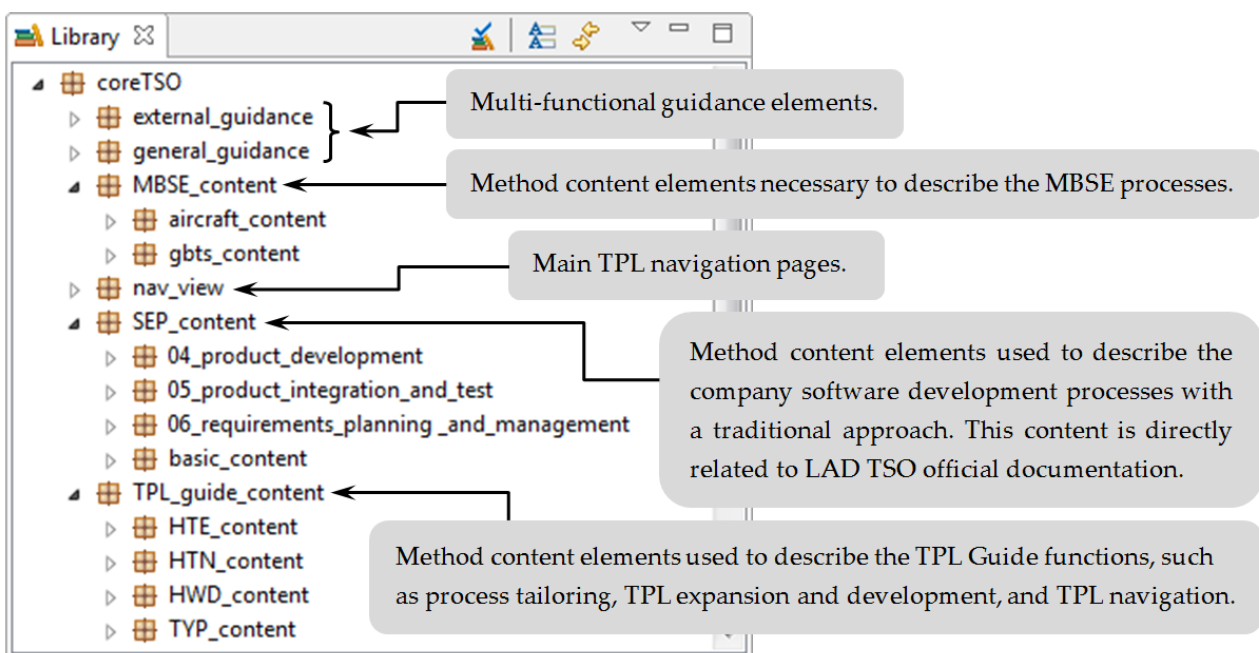


Figure 2.6 – ‘coreTSO’ module content

It can be noticed that only the content of three SEPs (product development, product integration and test, and requirements planning and management) was implemented inside TPL. The reason is that those are the Standard Enterprise Practices dedicated to the engineering process areas, thus necessary for the technical development of the tangible product.

- *MBSE_content*, that contains the method content elements exclusively necessary to describe product development processes with a MBSE approach. Note that most of the elements used to describe MBSE processes are the same used for processes featuring the standard approach; of course, they are not repeated, but referenced and recalled from ‘SEP-content’ plug-ins.
MBSE_content is sub-divided into aircraft content and ground-based training system content to enhance modularity and maintainability.
- *TPL_guide_content*, containing method content elements concerned with process tailoring (TYP), TPL expansion (HTE), TPL development (HWD) and TPL navigation guidelines (HTN).
- *external_guidance*, containing guidance elements coming from external sources, mainly

recycled from the Eclipse exemplary library ‘EPF Practices Library 1.5.1.5’.

- *general_guidance*, containing recursive guidance elements with general and multiple purpose.

processTSO

The plug-ins contained in this module must have references only toward ‘core’ plug-ins, that are necessary to complete processes with the involved method content elements. See paragraph ‘C.3 Plug-ins Logics’ and ‘2.5 Hierarchy of References’ for detailed information about this topic.

As illustrated in *Figure 2.7*, ‘processTSO’ module presents three main directories:

- *SEP_processes*, which contains processes and process elements featuring the company standard approach and are directly related to those described in OPAL documents. The two traditionally modelled delivery processes are within the ‘overall_processes’ directory, while sub-processes and smaller process elements are dislocated in Standard Enterprise Practices (SEPs) – thus process areas – dedicated plug-ins.
- *MBSE_processes*, that contains processes and process elements featuring a MBSE development approach. The two MBSE modelled de-

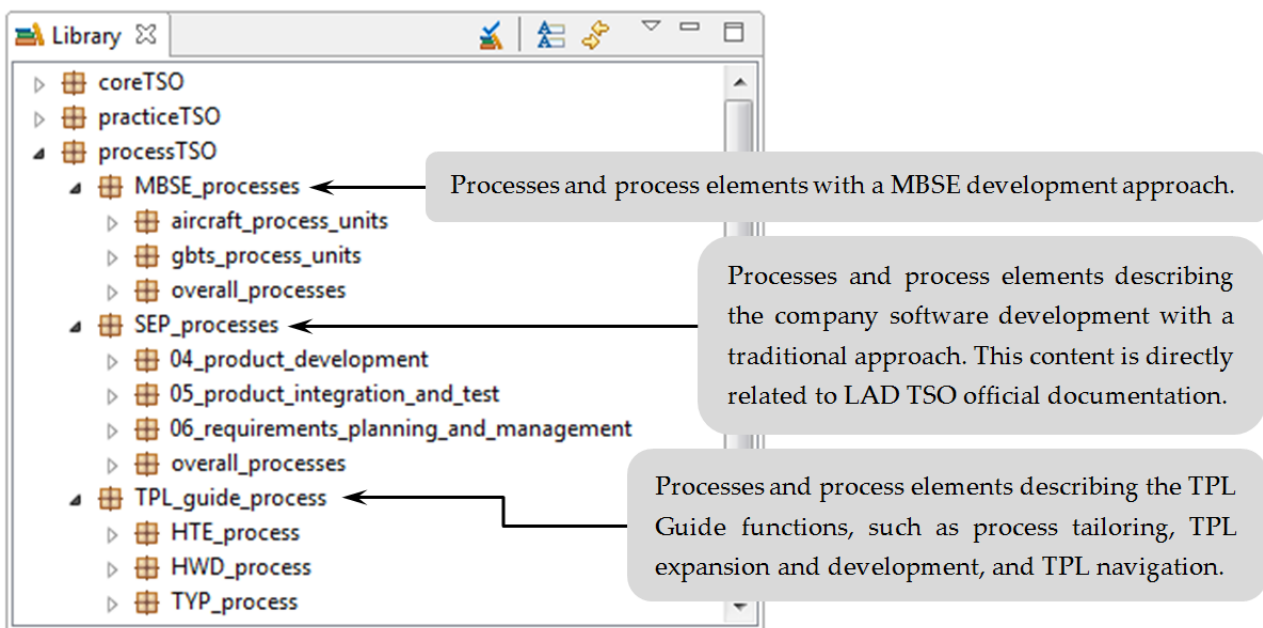


Figure 2.7 – ‘processTSO’ module content

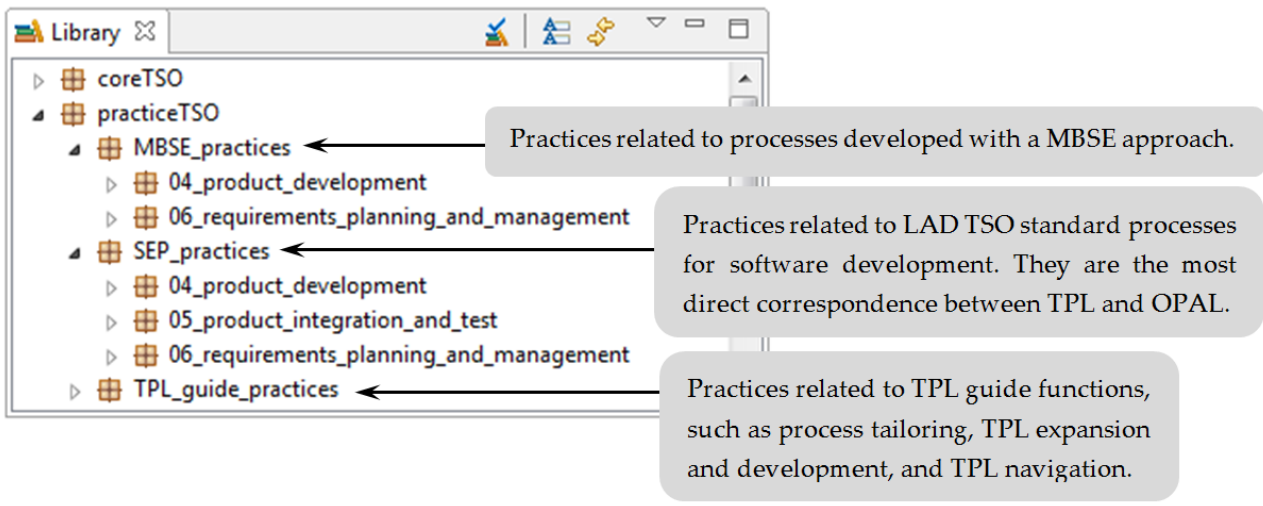


Figure 2.8 – ‘practiceTSO’ module content

livery processes are contained in the ‘*overall_processes*’ directory; most of the sub-processes and process elements used to build them are recalled by the ‘*SEP_processes*’ directory, while a few non-repeated process elements were created afresh in the ‘*aircraft_process_units*’ and ‘*gbts_process_units*’ directories.

- *TPL_guide_processes*, containing processes and process elements that describe process tailoring, TPL expansion and TPL navigation. For each of these functions there is a designated directory.

practiceTSO

In order to link framework content to practices, plug-ins contained in this module must have references toward ‘process’ and, if necessary, ‘core’ plug-ins (note that referencing a ‘process’ plug-in that in turn references a ‘core’ plug-in will make possible to recall both ‘process’ and ‘core’ content). See paragraph ‘*C.3 Plug-ins Logics*’ for detailed information about this EPF feature.

Figure 2.8 shows the ‘practiceTSO’ directories, mostly analogous to ‘processTSO’ ones:

- *SEP_practices*, that contains practices which offer a transverse view on company standard software development similarly to what the OPAL documentation does. In fact, the practices contained in this directory are the most direct correspondence between TPL and OPAL documents, since different pieces of content are organized into different plug-ins

depending on which Standard Enterprise Practice they are related to.

- *MBSE_practices*, that contains those practices that allow a MBSE version. TPL is specifically provided with product development and requirements management practices, that are those currently used by LAD. Procedures for product integration and test with a model-based approach do exist, but the current version of TPL does not feature them.
- *TPL_guide_practices*, which contains practices related to TPL Guide sections, namely process tailoring, TPL expansion and development – mostly sharing the same practices – and TPL navigation.

publishTSO

This module content has the main purpose of permitting TPL web-site navigation.

It mainly references ‘practice’ plug-ins, but also ‘core’ and ‘process’ ones whether necessary; again, see paragraph ‘*C.3 Plug-ins Logics*’ for detailed information about this topic.

Figure 2.9 shows the ‘publishTSO’ directories:

- *SEP_divided_practices*, containing custom categories with the function of practices clusters. In this directory, TPL practices are gathered depending on which OPAL part they are from, that is in which SEP their source documents can be located.

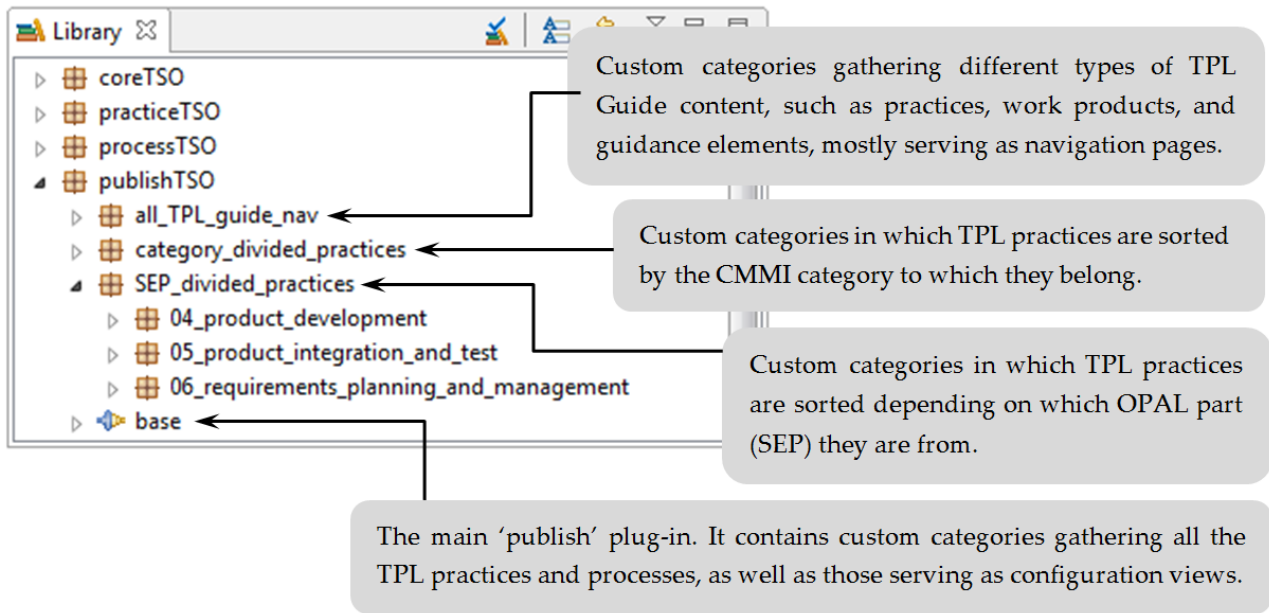


Figure 2.9 – 'publishTSO' module content

- *category_divided_practices*, that contain custom categories gathering the same practices gathered by the previous directory, but with a different order. In here, practices are sorted depending on which CMMI category their content belongs to.
- *base* plug-in content – illustrated in Figure 2.10 – consists of custom categories that in turn gather other custom categories. In TPL specific case these elements are used to gather the just introduced clusters of practices, all the TPL delivery processes, and all the TPL configuration views necessary to build up the web-site sidebar.
- *all_TPL_guide_nav*, that similarly to *base* plug-in contains custom categories used to gather and sort the different TPL Guide pieces of content and so to enhance the web-site navigability.

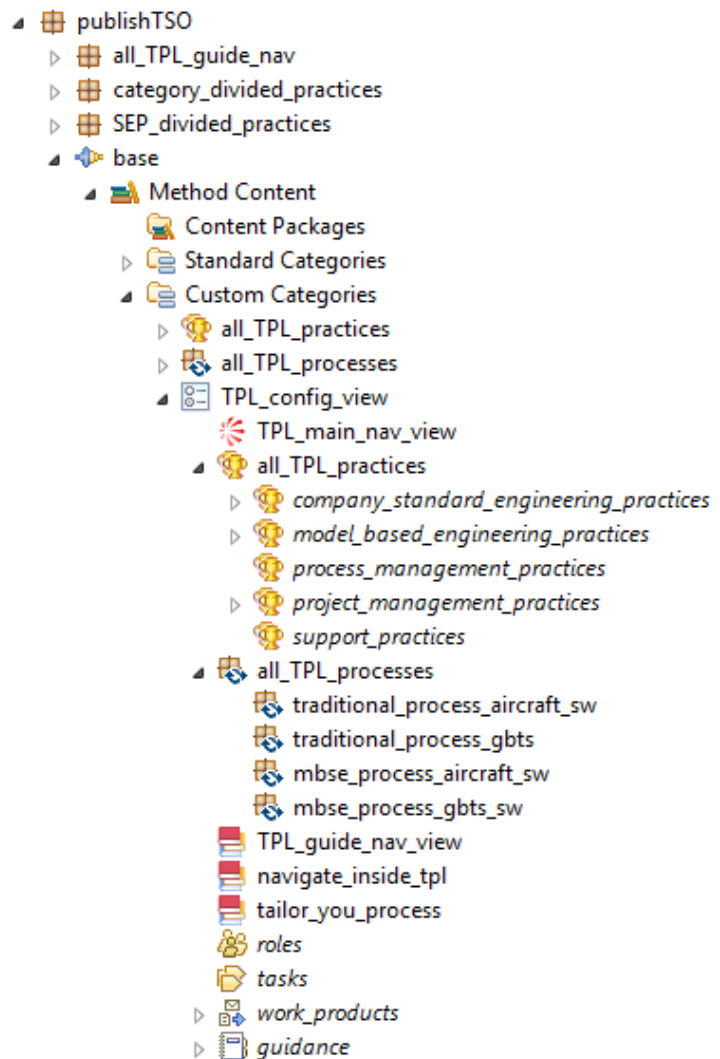


Figure 2.10 – Main 'publish' plug-in content

2.5 HIERARCHY OF REFERENCES

Alongside with the division of framework content into different directories, the hierarchy of plug-ins references was devel-

oped.

References are a crucial part of EPF Composer logics, and a proper management is indispensable to build a sustainable TSO Practices Library. Paragraph ‘C.3 Plug-ins Logics’ better explains these technical aspects of the software.

The hierarchy of references that characterizes TPL is sparsely outlined in the previous paragraph. Here it is explained in detail, also with the purpose of permitting a better comprehension of what lays behind its logics and the choices taken by the developers.

The first and fundamental set of criteria that rules the references chain of TPL is:

- a *publish* plug-in can have references toward any other kind of plug-in (*practice*, *process* and *core*);
- a *practice* plug-in can have references toward the *process* and, implicitly, the *core* kinds of plug-ins;
- a *process* plug-in can have references only toward the *core* kind of plug-in;

- a *core* plug-in cannot have references toward any other kind of plug-in.

Besides this, a plug-in can have a reference toward another one belonging to its same kind provided that no circular reference is generated.

Circular references are not allowed by EPF Composer for the clear reason that they would compromise the whole reference logics. Like stated in paragraph ‘C.3 Plug-ins Logics’, references are allowed as long as they have tree structure consisting of more reference chains intersected together.

The references ticked in the *publish* plug-ins present the biggest variability; some of them are used to gather method content elements and navigation pages located in *core* plug-ins, while others are used to gather practices and processes.

In Figure 2.11 is shown a scheme reporting an example of an allowed tree of references that roughly resembles TPL structure.

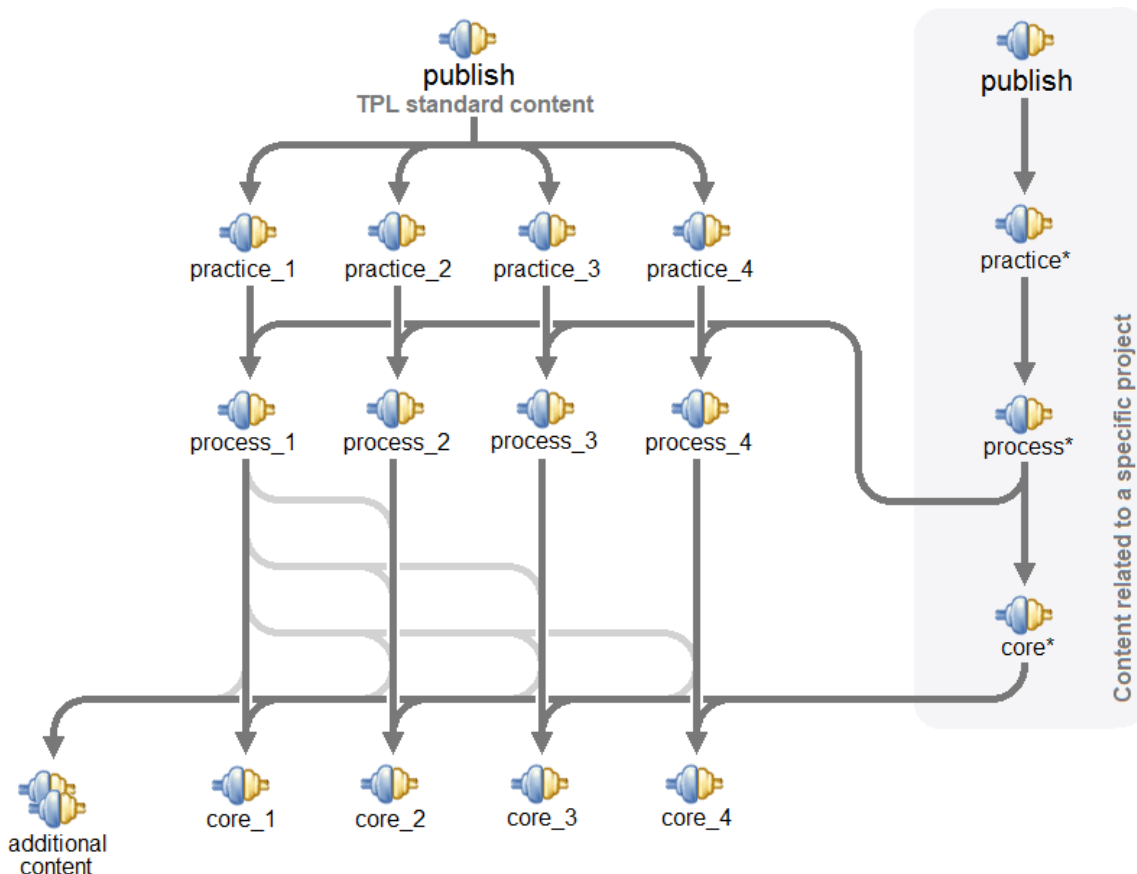


Figure 2.11 – Scheme of an allowed tree of references

2.6 LEVEL OF SPECIFICATION OF TPL CONTENT

TPL content developer has the delicate task of choosing the best specification level for newly created elements and processes.

Some factors condition this choice:

- usage frequency of an element,
- difference between a usage case and another,
- time required to specify it in the different usage cases.

The two general and opposite specification cases are reported hereafter to provide a better comprehension of the concept. A content developer who intends to create a general and versatile process should keep it at a rather low level of specification. Whenever a specification of that general process is required, the lowly-specified content is called up, customized and refined.

On the contrary, a developer that deals with a more specific process can adopt a higher level of specification, since the obtained process suits only for a few cases and a sharper explication of content is convenient.

2.7 EXPORTABILITY TO PROJECT MANAGEMENT SOFTWARE

IT market offers several tools for project management with support for project planning and

monitoring.

Some of them, such as Microsoft Project, permit to import processes modelled with EPF Composer so as to use the information they codify for project planning and project management.

Specifically, project management software such as Microsoft Project is designed to assist a project manager in developing a plan, assigning resources to tasks, tracking progress, managing the budget, and analysing workloads. With them it is also possible to optimize the project portfolio to prioritize initiatives and get the predefined results, as well as enabling organizations to proactively manage resource utilization, identify bottlenecks early, accurately forecast resource needs, and improve project selection and timely delivery.

Figure 2.12 shows the MBSE Process to design the Environmental Control System (ECS) of an Unmanned Aerial Vehicle (UAV) imported to Microsoft Project. The inherited sequence of activities can be integrated with resource deployment, worktime evaluations and budget aspects.

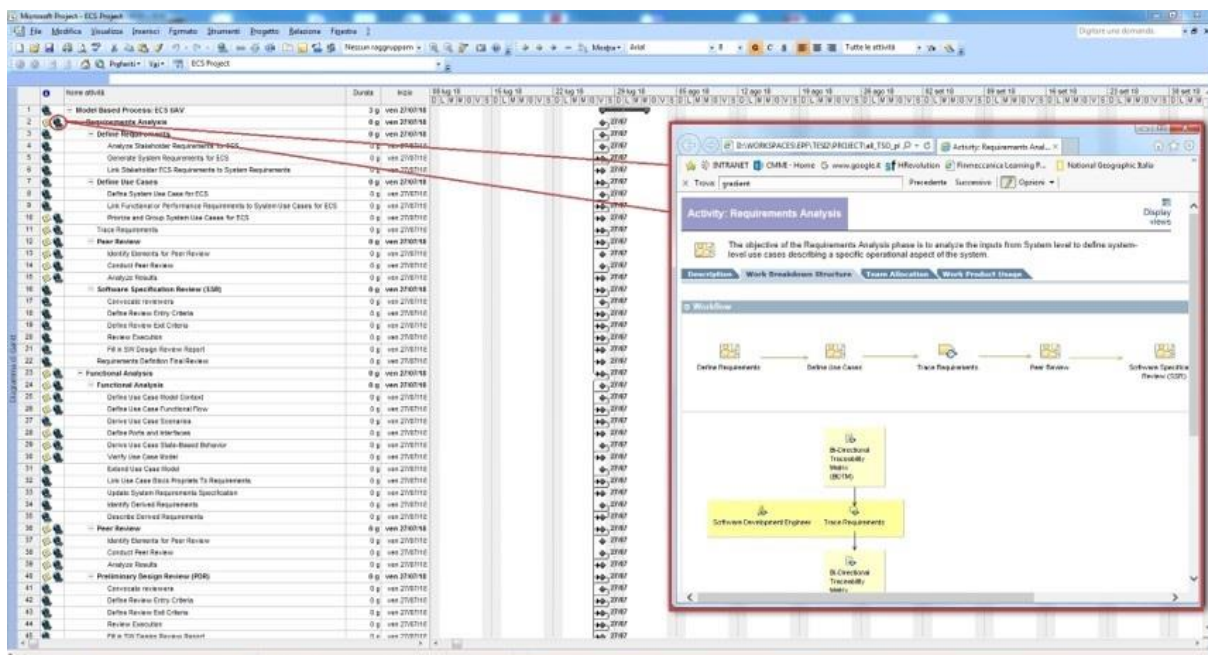


Figure 2.12 – Case study: ECS design process exported to Microsoft Project

Chapter 3

STUDY CONCLUSIONS: TECHNICAL AND ECONOMIC CONSIDERATIONS

3.1 COLLECTED EVIDENCE AND LESSONS LEARNED

The collected evidence and the lessons learned during the prototyping phase are a very important outcome of the ‘Feasibility Study’ project.

These elements, as illustrated in *Figure 3.1* and as already stated in *Chapter 1*, are the result of a critical analysis of prototyping activities and are a fundamental input to enhance TPL features and internal architecture.

Moreover, the lessons learned permitted to formulate the study conclusions explained in this chapter, in both their technical and economic aspects.

3.2 EVALUATION OF POSSIBLE BENEFITS

To properly evaluate the technical benefits and estimate the resulting economic impact is necessary

to look back at the project purposes.

In accordance to that, TPL users can actually benefit from a navigable knowledge-base of standard practices and processes; in detail, TPL offers:

- a top-down view on processes – and so a clear information about the sequence of activities – that can determine a considerable time saving, especially for what concerns the learning times for the youngest project engineers;
- a transversal view on processes by means of practices web-pages, organized similarly to the Standard Enterprise Practices of the document-based OPAL of TSO;
- the chance of reusing framework content, thus reducing ambiguities and avoiding the occurrence of doubts and misunderstandings.

For what concerns the ‘Tailor your Process’ feature of TPL, users can instantiate a standard process on

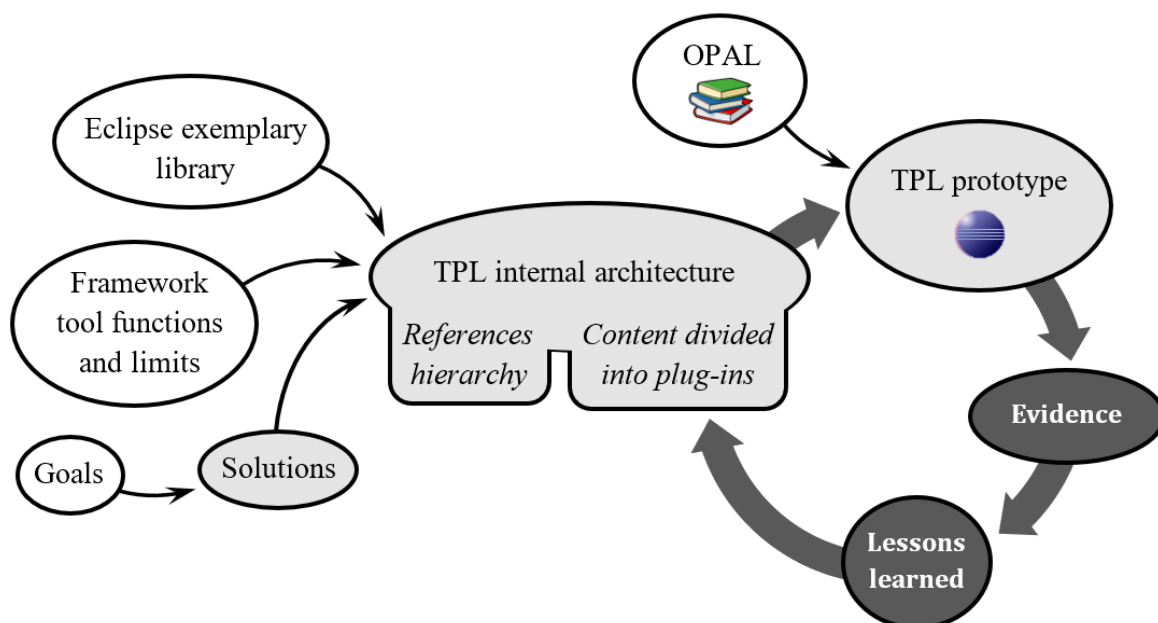


Figure 3.1 – Role of collected evidence and lessons learned in TPL development cycle

a specific project by following the proposed procedure.

Developing a process tailoring procedure was the final challenge of the internship project. Like stated before, there were not significant external prompts to direct the developers in this task, and besides satisfying the tailoring goals it was also necessary to be compliant with the already adopted TPL architecture and with EPF Composer functions and limits.

The developed tailoring procedure can boast of only one case study. Like shown in *Figure 3.2*, it consists in the instantiation of the *MBSE Process for A/C software development* on the project for designing the Environmental Control System (ECS) of an Unmanned Aerial Vehicle (UAV).

Implement a valid and efficient process tailoring feature inside TPL – alongside with the chance of reusing content – would bring the following advantages:

- defining a product development process in accordance with the customer’s needs and compliant with internal (TSO) and external (CMMI, Italian and European governments...) regulations would take a shorter time;
- all the defined processes adopted for different projects would have a good level of alignment and, consequently, the occurrence of ambiguities and doubtfulness would be reduced;
- EPF Composer offers the possibility to import a defined process into software for project management, such as Microsoft Project Manager®.

The last advantage that TPL presents when compared to the document-based OPAL regards maintenance. In fact, a well optimized library that is compliant with the content reuse and modularity concepts is easier to maintain, especially for what concerns keeping misunderstandings and ambiguities as rare as possible.

3.3 ESTIMATION OF COSTS

The higher cost involved in the development of a Company version of TPL is related to the required workforce.

Firstly, the appointed library manager should be sufficiently prepared and experienced in order to edit an optimally organized library. EPF Composer turned out to be a complicate and very delicate tool, with many functions to learn dealing with and, in case of inadequate usage, the risk of irreversible damage to the method library is considerable.

Secondly, implementing the knowledge-base and refining it till an acceptable level would require a certain amount of time split between a main process engineer and a supporting one. Evidences showed that employing more than two persons for directly working on the method library would create an excessive dispersion of content and facilitate the occurrence of severe problems.

Lastly, after TPL completion a designated role should take care of TPL aspects concerning its continuous maintenance, mainly consisting in:

- keeping TPL updated to the latest version of the Company practices, and
- supporting the internal customers that need to model a defined process based on their specific project.

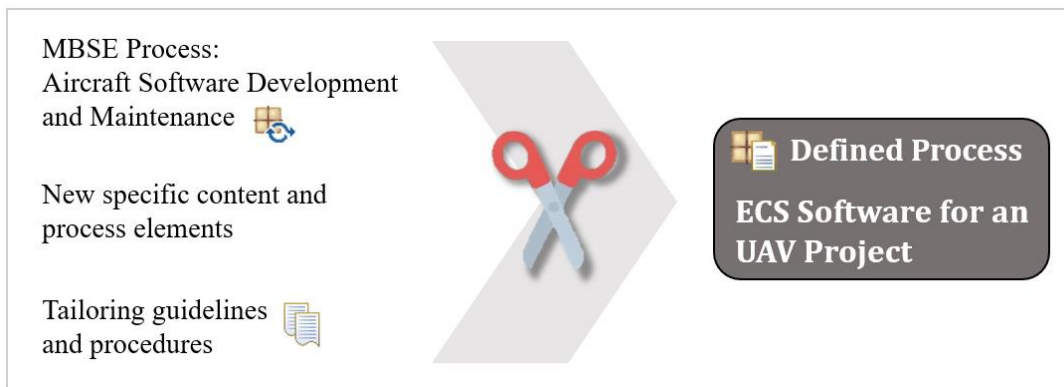


Figure 3.2 – Case Study of instantiation

Another cost to mention derives from the difficulties encountered by the users while navigating on the html web-site, since it tends to be quite dispersive. It was thought that implementing a set of good-navigation guidelines in TPL Guide might be useful for users in order to avoid losing their bearings.

However, like any difficulty that directly involves the tool, a good knowledge of EPF Composer and its 'Publish' function can make TPL web-site navigation much easier.

Considering the times and the efforts made to advance this project, a rough workload estimation can be done: the resources required to implement an effective and refined Company version of TPL could amount up to two man-years of work, also considering the time necessary to learn using EPF Composer and the time required to fully comprehend every piece of content of the LAD TSO document-based OPAL. Besides that, other resources are to be spent to continuously maintain and update TPL.

3.4 SUMMARY POINTS

A navigable and company-shared knowledge base featuring:

- reusable and modular content,
- practices and processes consultation,
- procedures and guidelines for process instantiation,

could bring the following benefits to the Training Systems Organization of LAD:

- i. comprehending the Company standard practices and processes would be easier, thanks to a more user-friendly and direct approach;
- ii. as a result, learning times for the youngest process engineers would be reduced;
- iii. it could be created a project-specific html web-site by instantiating a standard delivery process on the customer's project and publishing it, also with the possibility of using the existing pieces of framework content;
- iv. finally, a well optimized navigable knowledge base would be easier to maintain respect to a document-based one, since content repetition, dispersion and ambiguity would be greatly reduced.

On the other hand, creating, maintaining and using this kind of process engineering product has some disadvantages and costs:

- v. the html web-site navigation is not very easy and intuitive; the user must spend a little time dealing with it and, for the same reason, providing the knowledge base with good-navigation guidelines would be suggested;
- vi. to implement and maintain the knowledge base is required a skilled and proficient role, with a good knowledge of potentials and limits of the framework tool EPF Composer;
- vii. at present the complete and accurate implementation of the knowledge base would take up to two man-years of work, with a maximum of two employees working on it;
- viii. keeping the knowledge base continuously updated would take a considerable amount of time;
- ix. concerning the operative aspects, problems may easily arise while implementing the knowledge base due to its complexity and to EPF Composer management.

Appendix A

PRINCIPLES OF SYSTEMS ENGINEERING

A.1 INTRODUCTION

This appendix introduces the fundamental principles of systems engineering circumscribed to the topics faced within the ‘Feasibility Study’ project. The reported notions are not necessary to understand the present essay, but are fundamental to effectively approach TPL, that is the tangible product of the project. For this reason, it was thought that providing readers with the most general principles regarding the object of the study would have been useful to permit a full comprehension of every part of the project.

A.2 DEFINITIONS

A System is...

Simply stated, a system is an integrated composite of people, products, and processes that provide a capability to satisfy a stated need or objective.

Systems Engineering is...

Systems engineering consists of two significant disciplines: the technical knowledge domain in which the systems engineer operates, and systems engineering management.

Three commonly used definitions of systems engineering are provided by the best-known technical standards:

“A logical sequence of activities and decisions that transforms an operational need into a description of system performance parameters and a preferred system configuration. (MIL-STD 499A, Engineering Management, 1 May 1974. Now cancelled.)”

“An interdisciplinary approach that encompasses the entire technical effort, and evolves into and verifies an integrated and life cycle balanced set of system people, products, and process solutions that satisfy

customer needs. (EIA Standard IS-632, Systems Engineering, December 1994.)”

“An interdisciplinary, collaborative approach that derives, evolves, and verifies a life-cycle balanced system solution which satisfies customer expectations and meets public acceptability. (IEEE P1220, Standard for Application and Management of the Systems Engineering Process, [Final Draft], 26 September 1994.)”

In summary, systems engineering is an interdisciplinary engineering management process that evolves and verifies an integrated, life-cycle balanced set of system solutions that satisfy customer needs.

System Engineering Management is...

As illustrated in *Figure A.1*, systems engineering management is accomplished by integrating three major activities:

- *Development phasing*, that has two major purposes: it controls the design effort by developing design baselines that govern each level of development, and it interfaces with acquisition management by providing key events in the development process, where design viability can be assessed. The viability of the baselines developed is a major input for acquisition management Milestone (MS) decisions.
- *A systems engineering process*, that is the heart of systems engineering management. Its purpose is to provide a structured but flexible process that transforms requirements into specifications, architectures, and configuration baselines. The discipline of this process provides the control and traceability to develop solutions that meet customer needs.

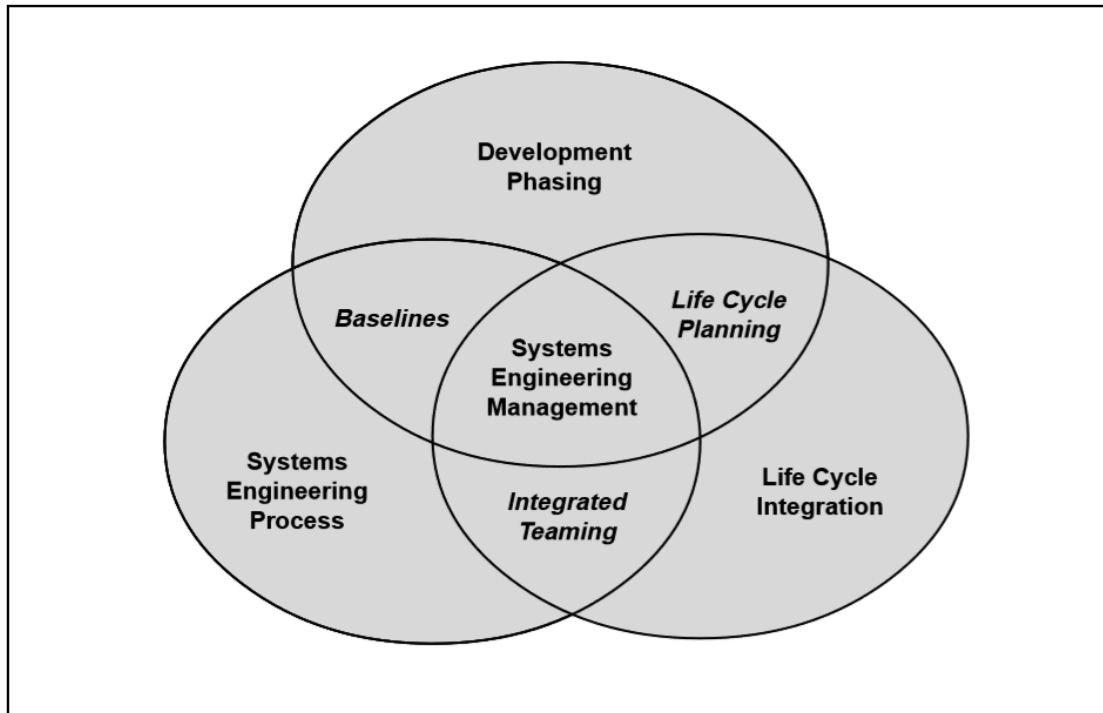


Figure A.1 – The three activities of systems engineering management

- *Life cycle integration*, necessary to ensure that the design solution is viable throughout the life of the system. It includes the planning associated with product and process development, as well as the integration of multiple functional concerns into the design and engineering process. In this manner, product cycle-times and the need for redesign and rework can be substantially reduced.

The themes introduced in this paragraph are very close to the topics of the ‘Feasibility Study’ project. In fact, the purpose of the framework tool EPF Composer consists in modelling the systems engineering processes in their fundamental aspects, that are the sequence of activities, who is in charge of performing them, what is required and produced by any different step, and the careful integration of all these elements through the whole product development process.

However, attention must be paid in order to distinguish between EPF Composer and systems engineering management purposes.

EPF Composer is a process engineering tool which permits to model and publish all the information necessary to systems engineers for a correct execution of product development processes.

Differently, Systems engineering management focuses on how to design and manage complex systems over their life cycle, and thus precedes the modelling activity carried out by a process engineering tool such as EPF Composer.

In the following paragraphs are explained in detail the three major activities of systems engineering management

A.3 DEVELOPMENT PHASING

Development usually progresses through distinct levels (or stages):

1. *concept level*, which produces a system concept description,
2. *system level*, which produces a system description in performance requirement terms,
3. *subsystem/component level*, which produces a product characteristics and performance description for each subsystem and component.

The systems engineering process is applied to each level of system development, one level at a time, to produce these descriptions commonly called *configuration baselines*. Baselines become more detailed with each level.

In the Department of Defence (DoD), the configuration baselines are called:

- *functional baseline* for the system-level description,
- *allocated baseline* for the subsystem/component performance descriptions, and
- *product baseline* for the subsystem/component detailed descriptions.

About Reviews and Audits...

A significant development at any given level in the system hierarchy should not occur until the configuration baselines at the higher levels are considered complete, stable, and controlled: reviews and audits are used to ensure that the baselines are ready for the next level of development. These review and audit processes also provide the necessary assessment of system maturity, which supports the DoD Milestone decision process. Also read the voice ‘*Verification*’ of paragraph ‘A.7 The Process in detail’.

A.4 THE SYSTEMS ENGINEERING PROCESS

The Systems Engineering Process is a top-down comprehensive, iterative and recursive problem solving process, applied sequentially through all

stages of development, that is used to transform needs and requirements into a set of system product and process descriptions (adding value and more detail with each level of development), as well as to generate information for decision makers and provide input for the next level of development.

As illustrated by *Figure A.2*, the fundamental systems engineering activities are:

- requirements analysis,
- functional analysis and allocation, and
- design synthesis,

all balanced by techniques and tools collectively called system analysis and control.

Systems engineering controls are used to track decisions and requirements, maintain technical baselines, manage interfaces, manage risks, track cost and schedule, track technical performance, verify requirements are met, and review/audit the progress.

During the systems engineering process architectures are generated to better describe and understand the system. The word ‘architecture’ is used in various contexts in the general field of engineering, however, Systems Engineering Management as developed in DoD recognizes three universally usable architectures that describe im-

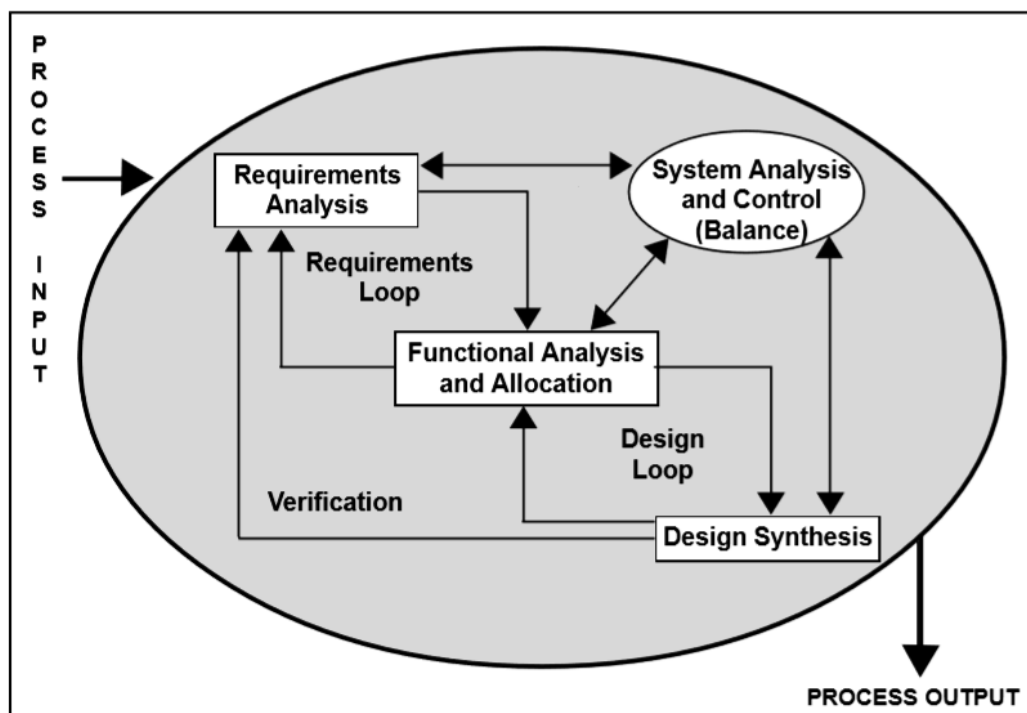


Figure A.2 – The systems engineering process

portant aspects of the system:

- *functional architecture* identifies and structures the allocated functional and performance requirements;
- *physical architecture* depicts the system product by showing how it is broken down into subsystems and components;
- *system architecture* identifies all the products that are necessary to support the system and all the processes necessary for development, production/construction, deployment, operations, support, disposal, training, and verification.

A.5 LIFE CYCLE INTEGRATION

Life cycle integration is achieved through integrated development, that is the concurrent consideration of all life cycle needs during the development process – activity that can be greatly enhanced through the use of interdisciplinary teams, often referred to as Integrated Product Teams (IPTs).

The objectives of an IPT are to produce a design solution that satisfies initially defined requirements and communicate that design solution clearly, effectively, and in a timely manner.

Life Cycle Functions

The eight primary life cycle functions are the characteristic actions associated with the system life cycle:

1. *Development* includes the activities required to evolve the system from customer needs to product or process solutions.
2. *Manufacturing/production/construction* includes the fabrication of engineering test models, low rate initial production and full-rate production of systems and/or end items.
3. *Deployment* (fielding) includes the activities necessary to initially deliver, assemble, install, checkout, train, operate, or field the system to achieve full operational capability.
4. *Operation* is the user function and includes activities necessary to satisfy defined operational objectives and tasks.
5. *Support* includes the activities necessary to provide operations support, maintenance, logistics, and material management.
6. *Disposal* includes the activities necessary to ensure that the disposal of system components meets all applicable regulations and directives.
7. *Training* includes the activities necessary to achieve and maintain the knowledge and skill levels necessary to efficiently and effectively perform operations and support functions.
8. *Verification* includes the activities necessary to evaluate progress and effectiveness of evolving system products and processes, measuring specification compliance.

These activities cover the ‘cradle to grave’ life cycle process. The system user’s needs are emphasized because they generate the requirement for the system, but it must be remembered that all of the life-cycle functional areas generate requirements for the systems engineering process once the user has established the basic needs.

A.6 MANDATORY RULES OF SYSTEMS ENGINEERING MANAGEMENT

DoD establishes two fundamental requirements for program management: it requires the adoption of an Integrated Product and Process approach, wherever practicable, and to use a disciplined systems engineering process to translate operational needs and/or requirements into a system solution.

Tailoring the Process

System engineering is applied during all acquisition and support phases for systems development and product improvements. Thus, the process must be tailored for different needs and/or requirements.

Tailoring considerations include system size and complexity, system definition detail level, scenarios and missions, constraints and requirements, technology base, major risk factors, and organizational best practices.

E.g., systems engineering of software should follow the basic systems engineering approach. However, it must be tailored to accommodate the software development environment, and the unique progress tracking and verification problems software development entails. In a like manner, all technology domains are expected to bring their own unique needs to the process.

Handbooks about the subject provide only a conceptual-level description of systems engineering management. Technical managers must tailor their systems engineering planning to meet their particular requirements and constraints, provided though that the basic time-proven concepts inherent in the systems engineering approach are retained to provide continuity and control.

A.7 THE PROCESS IN DETAIL

The Systems Engineering Process is, as already stated, a comprehensive, iterative and recursive problem solving process, applied sequentially top-down by integrated teams.

It transforms needs and requirements into a set of system product and process descriptions, generate information for decision makers, and provides input for the next level of development.

All the elements included in the Systems Engineering Process are shown by *Figure A.3* and

hereafter explained:

Systems Engineering Process Inputs

Inputs consist primarily of the customer's requirements, and project constraints. Inputs can include missions, measures of effectiveness, environments, available technology base, output requirements from prior application of the systems engineering process, program decision requirements, and requirements based on 'corporate knowledge'.

Requirements Analysis

The first step of the Systems Engineering Process is to analyse the process inputs. Requirements analysis is used to develop functional and performance requirements: customer requirements are translated into a set of requirements that define what the system must do and how well it must perform.

The systems engineer must ensure that the requirements are understandable, unambiguous, comprehensive, complete, and concise. Require-

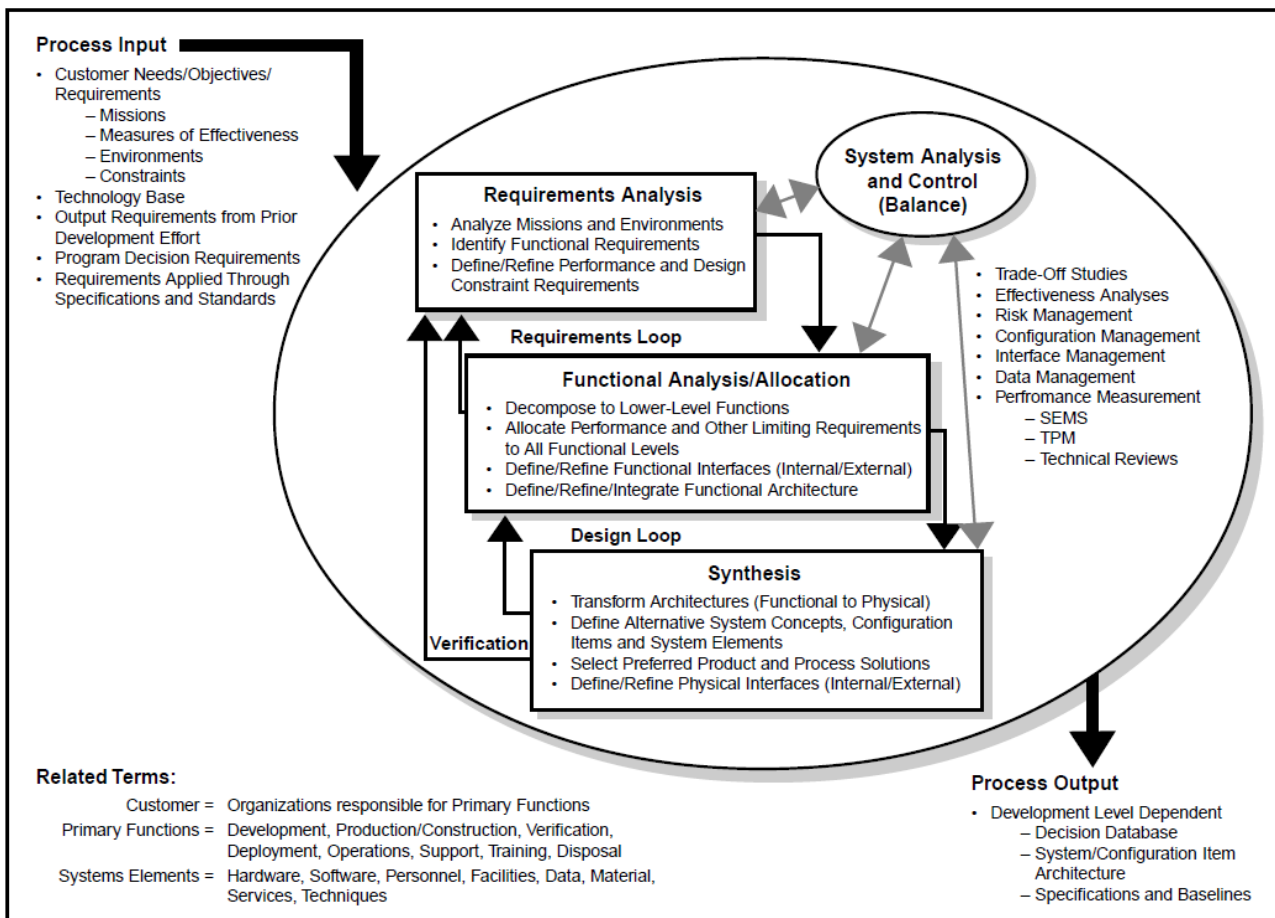


Figure A.3 – Detailed systems engineering process

ments analysis must clarify and define functional requirements and design constraints.

Functional Analysis/Allocation

Functional analysis is performed decomposing higher-level functions – identified through requirements analysis – into lower-level functions. The performance requirements associated with the higher-level are allocated to lower functions. The result is a description of the product or item in terms of what it does logically and in terms of the performance required. This description is often called the functional architecture of the product. Functional analysis and allocation permit to better understand what the system has to do, in what ways it can do it, and the priorities and conflicts associated with lower-level functions. Key tools in functional analysis and allocation are functional flow block diagrams, time line analysis, and the requirements allocation sheet.

Requirements Loop

Performing the functional analysis and allocation gives a better understanding of the requirements and should prompt a reconsideration of the re-

quirements analysis.

Each function identified should be traceable back to a requirement. This iterative process of revisiting requirements analysis as a result of functional analysis and allocation is referred to as the requirements loop.

Design Synthesis

Design synthesis is the process of defining the product or item in terms of the physical and software elements which together make up and define the item. The result is often referred to as the physical architecture and is the basic structure for generating the specifications and baselines.

Design Loop

Similar to the requirements loop described above, the design loop is the process of revisiting the functional architecture to verify that the physical design synthesized can perform the required functions at required levels of performance. The design loop permits reconsideration of how the system will perform its mission, and this helps optimize the synthesized design.

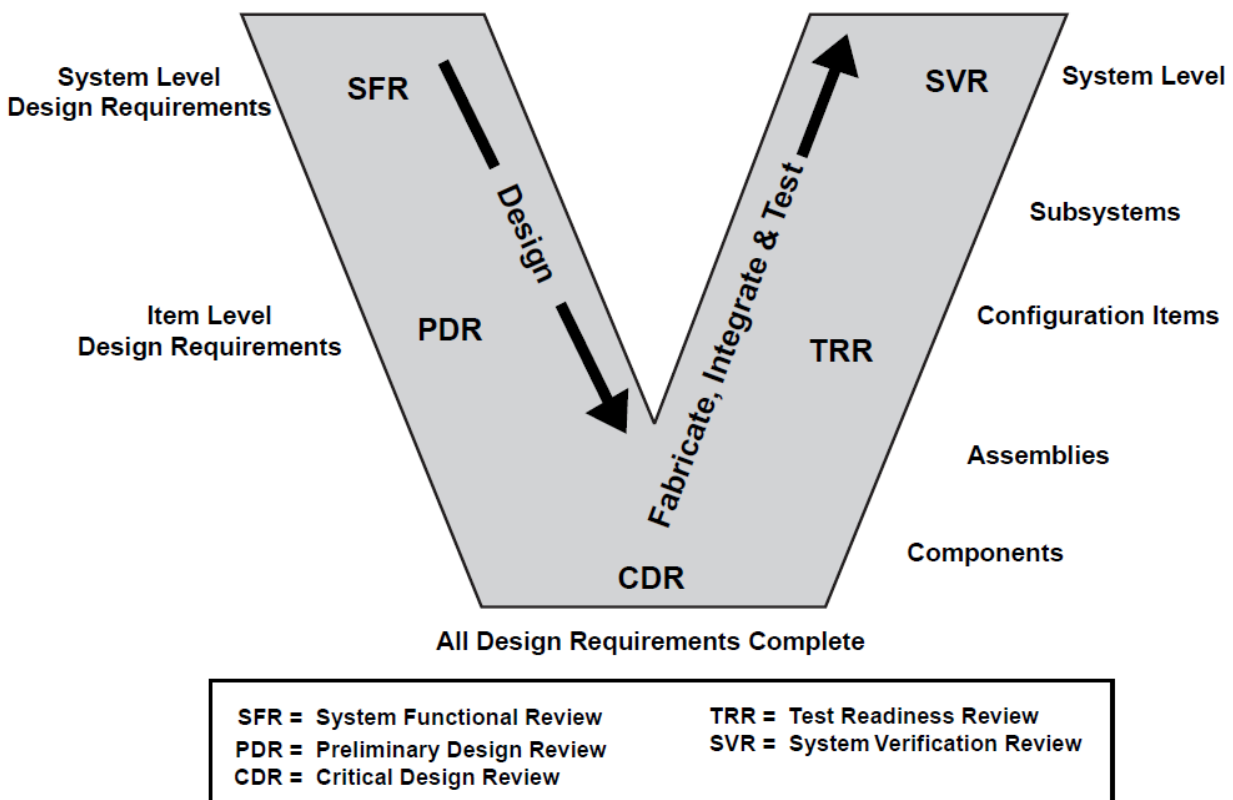


Figure A.4 – Systems engineering and verification

Verification

For each application of the system engineering process, the solution will be compared to the requirements. This part of the process is called the verification loop, or more commonly, Verification. Each requirement at each level of development must be verifiable.

Baseline documentation developed during the systems engineering process must establish the method of verification for each requirement. Appropriate methods of verification include examination, demonstration, analysis (including modelling and simulation), and testing. Formal test and evaluation (both developmental and operational) are important contributors to the verification of systems. As reported in *Figure A.4*, the different verification steps depend on the advancement level of the project. The first three steps are conducted during the design phase:

- system functional review,
 - preliminary design review,
 - critical design review,
- and they are followed by the operational testing:
- test readiness review,
 - system verification review.

Systems Analysis and Control

Systems Analysis and Control include technical management activities required to measure progress, evaluate and select alternatives, and document data and decisions. These activities apply to all steps of the systems engineering process.

System analysis activities include trade-off studies, effectiveness analyses, and design analyses. They evaluate alternative approaches to satisfy technical requirements and program objectives, and provide a rigorous quantitative basis for selecting performance, functional, and design requirements.

Control activities include risk management, configuration management, data management, and performance-based progress measurement including event-based scheduling, Technical Performance Measurement (TPM), and technical reviews.

Systems Engineering Process Output

Process output is dependent on the level of development. It will include the decision database, the system or configuration item architecture, and the baselines, including specifications, appropriate to the phase of development.

In general, it is any data that describes or controls the product configuration or the processes necessary to develop that product.

A.8 SUMMARY POINTS

- i. Systems engineering management is a multifunctional process that integrates life cycle functions, the systems engineering problem-solving process, and progressive baselining.
- ii. Integrated Product Teams should apply the systems engineering process to develop a life cycle balanced-design solution.
- iii. The systems engineering process is applied to each level of development, one level at a time.
- iv. Fundamental systems engineering activities are requirements analysis, functional analysis/allocation, and design synthesis, all of which are balanced by system analysis and control activities.
- v. Baseline phasing provides for an increasing level of descriptive detail of the products and processes with each application of the systems engineering process. In fact, baselining is a nut-shell concept, in which a first system definition leads to component definitions, and then to component designs, that finally lead to a product.
- vi. The output of each application of the systems engineering process is a major input to the next process application.

Appendix B

INTRODUCTION TO THE CMMI® INSTITUTE DEVELOPMENT MODEL

B.1 INTRODUCTION

This appendix introduces the fundamental concepts developed by the CMMI Institute. The reported notions are not necessary to understand the present essay, but provide a useful background to develop TPL in accordance with the CMMI model adopted by the Training Systems Organization of LAD.

B.2 CAPABILITY MATURITY MODELS

A Capability Maturity Model® (CMM®), including CMM Integration, is a simplified representation of the world. CMMs focus on improving processes in an organization. They contain the essential elements of effective processes for one or more disciplines and describe an evolutionary improvement path from ad hoc, immature processes to disciplined, mature ones with improved quality and effectiveness.

Like other CMMs, CMMI models provide guidance to use when developing processes. CMMI models are not processes or process descriptions; the actual processes used in an organization depend on many factors, including application domains and organization structure and size. In particular, the process areas of a CMMI model typically do not map one to one with the processes used in the organization of a specific Company.

CMMI for Development (CMMI-DEV) consists of best practices that address development activities applied to products and services. It addresses practices that cover the product's lifecycle from conception through delivery and maintenance. The emphasis is on the work necessary to build and maintain the total product

B.3 PROCESS AREA COMPONENTS

Model components are grouped into three categories – required, expected, and informative – that reflect how to interpret them.

Required components are CMMI components that are essential to achieving process improvement in a given process area. This achievement must be visibly implemented in an organization's processes. Goal satisfaction is used in appraisals as the basis for deciding whether a process area has been satisfied.

A Company adopting the CMMI model must present the required components in order to institutionalize process areas; as long as one of the goals for a certain maturity level is not achieved the corresponding level cannot be considered as reached.

Expected components describe the activities that are important in achieving a required CMMI component. Expected components guide those who implement improvements or perform appraisals.

Before goals can be considered to be satisfied, either their practices as described, or acceptable alternatives to them, must be present in the planned and implemented processes of the organization.

In other words, a Company is allowed to achieve the goals using activities different from the ones described in the practices, as long as they satisfy the need of achieving the goals

Informative components are neither expected nor required. These components can be example boxes, detailed explanations, or other helpful information. Sub-practices, notes, references, goal titles, practice titles, sources, example work products, and generic practice elaborations are informative model components.

The informative material plays an important role in understanding the required and expected model

components. It is often impossible to adequately describe the behaviour required or expected of an organization using only a single goal or practice statement. The model's informative material provides information necessary to achieve the correct understanding of goals and practices and thus cannot be ignored.

The components found in each process area and in the generic goals and generic practices are summarized in *Figure B.1* to illustrate their relationships.

Process Areas

A process area is a required component consisting of a cluster of related practices in an area that, when implemented collectively, satisfies a set of goals considered important for making improve-

ment in that area.

CMMI models divide practices in a total of 22 process areas; 16 of them are known as core process areas, since they cover basic concepts that are fundamental to process improvement in any area of interest (i.e., acquisition, development, services). Some of the material in the core process areas is the same in all constellations, while other material may be adjusted to address a specific area of interest. Consequently, the material in the core process areas may not be exactly the same.

Purpose Statements

A purpose statement describes the purpose of the process area.

Introductory Notes

The introductory notes section of the process area

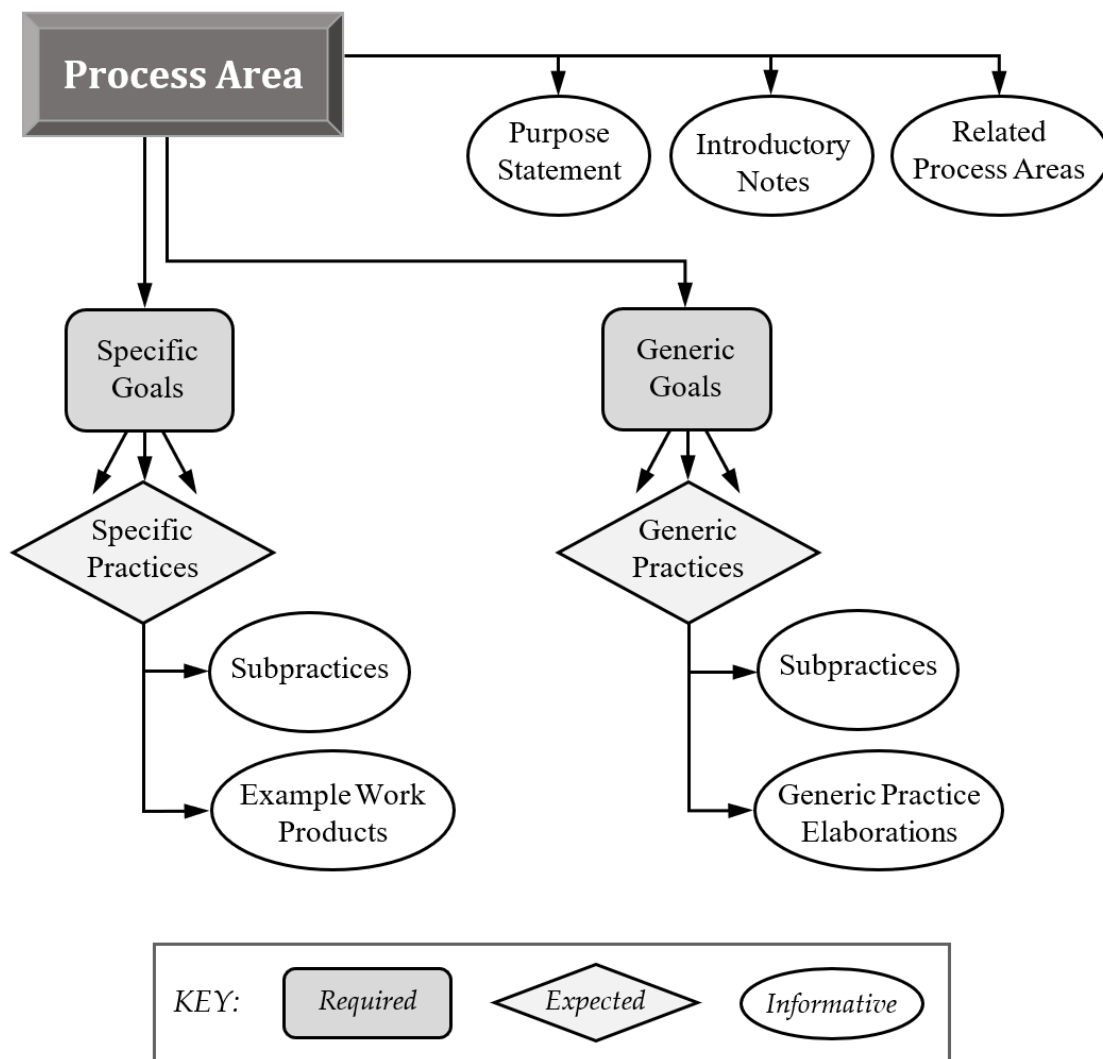


Figure B.1 – CMMI model components

describes the major concepts covered in the process area. An example from the introductory notes of the Project Monitoring and Control process area is *‘When actual status deviates significantly from expected values, corrective actions are taken as appropriate’*.

Related Process Areas

The related process areas section reflects the high-level relationships among the process areas. Related process areas section is an informative component.

An example of a reference found in the Related Process Areas section of the Project Planning process area is *‘Refer to the Risk Management process area for more information about identifying and analysing and mitigating risks’*.

Specific Goals

A specific goal describes the unique characteristics that must be present to satisfy the process area. A specific goal is a required model component and is used in appraisals to help determine whether a process area is satisfied.

For example, a specific goal from the Configuration Management process area is *‘Integrity of baselines is established and maintained’*.

Only the statement of the specific goal is a required model component. Goal title and notes are considered informative model components.

Specific Practices

A specific practice is the description of an activity that is considered important in achieving the associated specific goal. A specific practice is an expected model component.

For example, a specific practice from the Project Monitoring and Control process area is *‘Monitor commitments against those identified in the project plan’*.

Only the statement of the specific practice is an expected model component. The title of a specific practice (preceded by the practice number) and notes associated with the specific practice are considered informative model components.

Example Work Products

The example work products section lists sample outputs from a specific practice. An example work product is an informative model component.

Subpractices

A subpractice is a detailed description that provides guidance for interpreting and implementing a specific or generic practice. Subpractices are an informative component meant only to provide ideas that may be useful for process improvement.

Generic Goals

Generic goals are called ‘generic’ because the same goal statement applies to multiple process areas. A generic goal describes the characteristics that must be present to institutionalize processes that implement a process area. A generic goal is a required model component and is used in appraisals to determine whether a process area is satisfied. An example of a generic goal is *‘The process is institutionalized as a managed process’*.

Only the statement of the generic goal is a required model component. Goal title and notes are considered informative model components.

Generic Practices

Generic practices are called ‘generic’ because the same practice applies to multiple process areas. The generic practices associated with a generic goal describe the activities that are considered important in achieving the generic goal and contribute to the institutionalization of the processes associated with a process area. A generic practice is an expected model component.

For example, a generic practice for the generic goal *‘The process is institutionalized as a managed process’* is *“Provide adequate resources for performing the process, developing the work products, and providing the services of the process”*.

Only the statement of the generic practice is an expected model component.

Generic Practice Elaborations

Generic practice elaborations appear after generic practices to provide guidance on how the generic practices can be applied uniquely to process areas. A generic practice elaboration is an informative model component.

B.4 CAPABILITY AND MATURITY LEVELS

Levels are used in CMMI-DEV to describe an evolutionary path recommended for an organiza-

tion that wants to improve the processes it uses to develop products or services. Levels can also be the outcome of the rating activity in Company appraisals. Appraisals can apply to entire organizations or to smaller groups such as a group of projects or a division.

CMMI supports two improvement paths based on levels, that in turn are associated with a type of level and an approach to process improvement called ‘representation’ (see *Figure B.2* to better understand the two different representations):

The first path enables organizations to incrementally improve processes corresponding to an individual process area (or group of process areas) selected by the organization. This path is associated to *capability levels* and *continuous representations* to achieve them.

The second path enables organizations to improve a set of related processes by incrementally addressing successive sets of process areas. This path is associated with *maturity levels* and *staged representations* to achieve them.

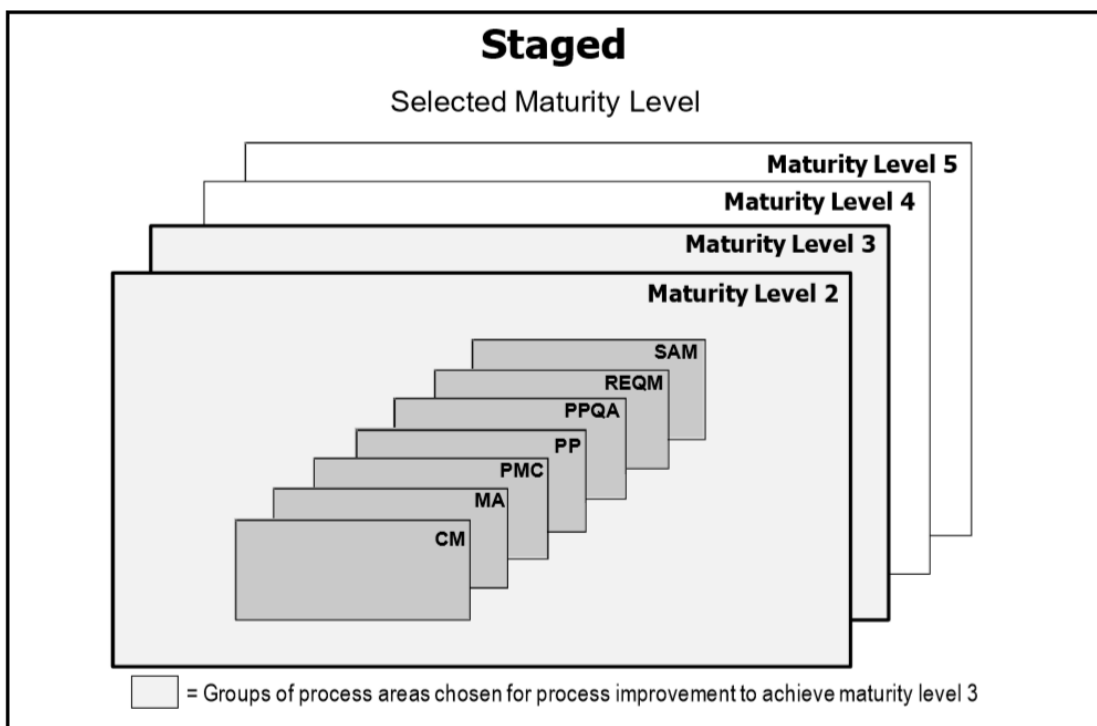
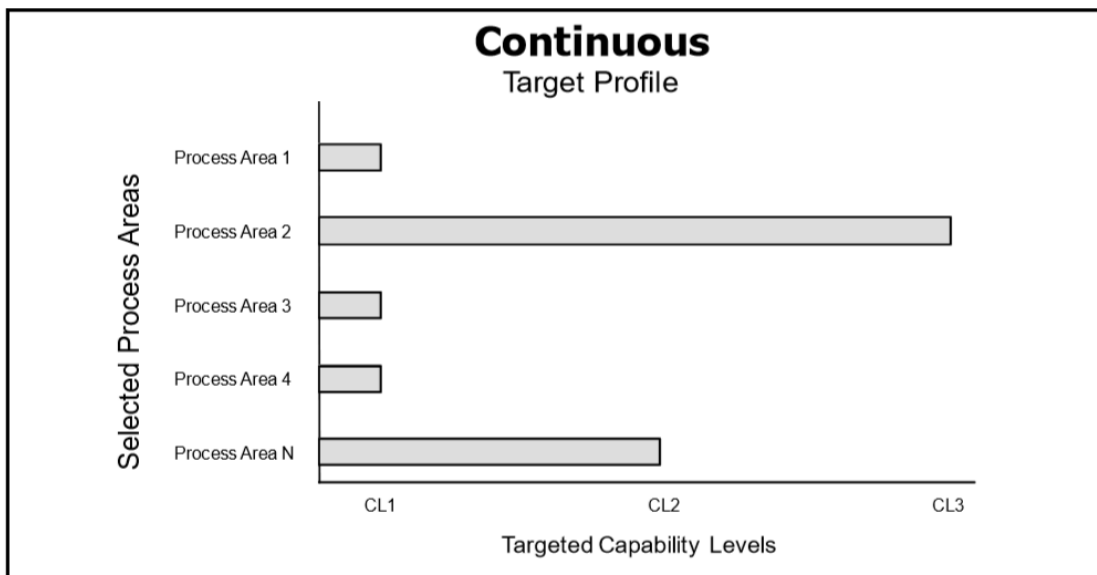


Figure B.2 – Process areas in the continuous and staged representations

The LAD TSO managed to reach both capability and maturity level 3. It can be observed that TPL goals and features reflect the level 3 concepts hereafter explained.

Capability Level 3: Defined

A capability level 3 process is characterized as a defined process. A defined process:

- is a managed process that is tailored from the organization's set of standard processes according to the organization's tailoring guidelines;
- has a maintained process description;
- contributes process related experiences to the organizational process assets.

A critical distinction between capability levels 2 and 3 is the scope of standards, process descriptions, and procedures. At capability level 2, the standards, process descriptions, and procedures can be quite different in each specific instance of the process (e.g., on a particular project). At capability level 3, the standards, process descriptions, and procedures for a project are tailored from the organization's set of standard processes to suit a particular project or organizational unit and therefore are more consistent, except for the differences allowed by the tailoring guidelines.

Another critical distinction is that at capability level 3 processes are typically described more rigorously than at capability level 2. A defined process clearly states the purpose, inputs, entry criteria, activities, roles, measures, verification steps, outputs, and exit criteria.

At capability level 3, processes are managed more proactively using an understanding of the interrelationships of the process activities and detailed measures of the process and its work products.

Maturity Level 3: Defined

At maturity level 3, processes are well characterized and understood, and are described in standards, procedures, tools, and methods. The organization's set of standard processes, which is the basis for maturity level 3, establishes consistency across the organization and is improved over time. Projects establish their defined processes by tailoring the organization's set of standard processes according to tailoring guidelines.

Similarly to what occurs at capability level 3, at maturity level 3 the standards, process descriptions, and procedures for a project are tailored from the organization's set of standard processes

to suit a particular project or organizational unit and therefore are more consistent except for the differences allowed by the tailoring guidelines.

Another critical distinction is that at maturity level 3, processes are typically described more rigorously than at maturity level 2. A defined process clearly states the purpose, inputs, entry criteria, activities, roles, measures, verification steps, outputs, and exit criteria.

At maturity level 3, processes are managed more proactively using an understanding of the interrelationships of process activities and detailed measures of the process, its work products, and its services.

Finally, at maturity level 3 the organization further improves its processes that are related to the maturity level 2 process areas. Generic practices associated with generic goal 3 that were not addressed at maturity level 2 are applied to achieve maturity level 3.

B.5 PROCESS AREAS

Process areas are viewed differently in the two representations. *Figure B.2* compares views of how process areas are used in the continuous representation and the staged representation.

The *continuous representation* enables the organization to choose the focus of its process improvement efforts by choosing those process areas, or sets of interrelated process areas, that best benefit the organization and its business objectives. Although there are some limits on what an organization can choose because of the dependencies among process areas, the organization has considerable freedom in its selection.

To support those who use the continuous representation, process areas are organized into four categories:

1. Process Management,
2. Project Management,
3. Engineering, and
4. Support.

These categories emphasize some of the key relationships that exist among the process areas.

Once process areas are selected, the desired and appropriate capability level must be selected, so to define how mature the processes associated with those process areas should become.

The selection of a combination of process areas and capability levels is typically described in a 'target profile'. A target profile defines all of the process areas to be addressed and the targeted capability level for each. This profile governs which goals and practices the organization will address in its process improvement efforts.

Organizations that target capability levels higher than 1 concentrate on the institutionalization of selected processes in the organization by implementing generic goals and practices.

The *staged representation* provides a path of improvement from maturity level 1 to maturity level 5 that involves achieving the goals of the process areas at each maturity level. To support those who use the staged representation, process areas are grouped by maturity level, indicating which process areas to implement to achieve each maturity level.

and capability levels is typically described in a 'target profile'. A target profile defines all of the process areas to be addressed and the targeted capability level for each. This profile governs which goals and practices the organization will address in its process improvement efforts.

Organizations that target capability levels higher than 1 concentrate on the institutionalization of selected processes in the organization by implementing generic goals and practices.

Figure B.3 provides a table of CMMI-DEV process areas and their associated categories and maturity levels.

B.6 PROCESS IMPROVEMENT PROGRAM

Three selections must be made to apply CMMI to an organization for process improvement:

1. selecting a part of the organization;
2. selecting a model;
3. selecting a representation.

When selecting the projects to involve in process improvement program the available resources should be specified and considered in order to not overcharge the effort. The selection should also consider organizational, product, and work homogeneity.

Selecting an appropriate model is also essential to a successful process improvement program. For example, the CMMI-DEV model focuses on activities for developing quality products and services. The CMMI-ACQ model focuses on activities for initiating and managing the acquisition of products and services. The CMMI-SVC model focuses on activities for providing quality services to the customer and end users. When selecting a model, appropriate consideration should be given to the primary focus of the organization and projects, as well as to the processes necessary to satisfy business objectives.

The selected representation (capability or maturity levels) must fit the current concept of process improvement. Regardless of which representation is chosen, nearly any process area or group of process areas can be selected to guide improvement, although dependencies among process areas should be considered when making such a selection.

As process improvement plans and activities progress, other important selections must be made, including whether to use an appraisal, which appraisal method should be used, which projects should be appraised, how training for staff should be secured, and which staff members should be trained.

B.7 CMMI MODELS

CMMI models describe best practices that organizations have found to be productive and useful to achieving their business objectives. Regardless of the organization, professional judgment must be used when interpreting CMMI best practices for specific situation, needs, and business objectives. This use of judgment is reinforced when words such as 'adequate', 'appropriate' or 'as needed' appear in a goal or practice. These words are used for activities that may not be equally relevant in all situations thus goals and practices must be interpreted in ways that specifically work for the organization.

As a customer begins using a CMMI model to improve the processes of an organization, real world processes should be mapped to CMMI process areas. This mapping enables the initial judgement and the later tracking of the level of

<i>Process Area</i>	<i>Category</i>	<i>Maturity Level</i>
Causal Analysis and Resolution (CAR)	Support	5
Configuration Management (CM)	Support	2
Decision Analysis and Resolution (DAR)	Support	3
Integrated Project Management (IPM)	Project Management	3
Measurement and Analysis (MA)	Support	2
Organizational Process Definition (OPD)	Process Management	3
Organizational Process Focus (OPF)	Process Management	3
Organizational Performance Management (OPM)	Process Management	5
Organizational Process Performance (OPP)	Process Management	4
Organizational Training (OT)	Process Management	3
Product Integration (PI)	Engineering	3
Project Monitoring and Control (PMC)	Project Management	2
Project Planning (PP)	Project Management	2
Process and Product Quality Assurance (PPQA)	Support	2
Quantitative Project Management (QPM)	Project Management	4
Requirements Development (RD)	Engineering	3
Requirements Management (REQM)	Project Management	2
Risk Management (RSKM)	Project Management	3
Supplier Agreement Management (SAM)	Project Management	2
Technical Solution (TS)	Engineering	3
Validation (VAL)	Engineering	3
Verification (VER)	Engineering	3

Figure B.3 – Table of process areas, categories, and maturity levels

conformance of the organization to the adopted CMMI model. Opportunities for improvement are also identified thanks to mapping.

To interpret practices, it is important to consider the overall context in which these practices are used and to determine how well the practices satisfy the goals of a process area in that context. CMMI models do not prescribe nor imply processes that are right for any organization or project. Instead, CMMI describes minimal criteria necessary to plan and implement processes selected by the organization for improvement based on business objectives.

CMMI practices purposely use nonspecific phrases such as ‘relevant stakeholders’, ‘as appropriate’ and ‘as necessary’ to accommodate the needs of different organizations and projects, that also could differ at various points of the life of project.

B.8 PROCESS INSTITUTIONALIZATION

Institutionalization is an important concept in process improvement. When mentioned in the generic goal and generic practice descriptions, institutionalization implies that the process is ingrained

in the way the work is performed and there is commitment and consistency to performing (i.e., executing) the process.

An institutionalized process is more likely to be retained during times of stress. When the requirements and objectives for the process change, however, the implementation of the process may also need to change to ensure that it remains effective. The generic practices describe activities that address these aspects of institutionalization.

The degree of institutionalization is embodied in the generic goals and expressed in the names of the processes associated with each goal.

Performed Process

A performed process is a process that accomplishes the work necessary to satisfy the specific goals of a process area.

Managed Process

A managed process is a performed process that is planned and executed in accordance with policy, employs skilled people having adequate resources to produce controlled outputs and involves relevant stakeholders is monitored. Moreover, it is controlled, reviewed and is evaluated for adherence to its process description.

A critical distinction between a performed process and a managed process is the extent to which the process is managed. A managed process is planned and the execution of the process is managed against the plan.

Defined Process

A defined process is a managed process that is tailored from the organization's set of standard processes according to the organization's tailoring guidelines; has a maintained process description; and contributes process related experiences to the organizational process assets.

A critical distinction between a managed process and a defined process is the scope of application of the process descriptions, standards, and procedures. For a managed process, the process descriptions, standards, and procedures are applicable to a particular project, group, or organizational function. As a result, the managed processes of two projects in one organization can be different.

B.9 ORGANIZATIONAL PROCESS ASSETS

Organizational process assets are artifacts that relate to describing, implementing, and improving processes. These artifacts are assets because they are developed or acquired to meet the business objectives of the organization and they represent investments by the organization that are expected to provide current and future business value.

The organizational process assets developed by an organization are collected in its Organization's Process Asset Library (OPAL).

OPAL is a library of information used to store and make process assets that are useful to those who are defining, implementing, and managing processes in the organization. This library contains process assets that include process related documentation such as policies, defined processes, checklists, lessons learned documents, templates, standards, procedures, plans, and training materials.

Appendix C

INTRODUCTION TO EPF COMPOSER

C.1 ABOUT THE EPF PROJECT

The Eclipse Process Framework (EPF) project aims at producing a customizable software process engineering framework, with exemplary process content and tools, supporting a broad variety of project types and development styles.

Method content and processes are structured based on a formal meta-model. The initial version of this meta-model has been derived from IBM's Unified Method Architecture (UMA), that is an evolution of the current OMG (Object Management Group) industry standard Software Process Engineering Meta-model (SPEM) v1.1 integrating concepts from IBM Rational Unified Process, IBM Global Services, and IBM Rational Summit Ascendant.

IBM and other OMG partners are working on making UMA, with improvements suggested by partners, to become SPEM 2.0. The initial exemplary tool implementation for EPF will be based on the first draft submission. As SPEM 2.0 stabilizes, it is expected to update the EPF to the final specification. The meta-model will be extensible

through the usage of custom attributes and custom process elements as well as normal Eclipse Modeling Framework (EMF) extensibility mechanisms.

C.2 EXEMPLARY TOOL: EPF COMPOSER

The EPF Composer is a free, open-source tool platform for enterprise architects, programme managers, process engineers, project leads and project managers to implement, deploy, and maintain processes for organisations or individual projects.

The tool helps development professionals set up a knowledge base of intellectual capital that lets them browse, manage and deploy content. This content can be licensed, acquired, or – perhaps most importantly – developed in house. It can comprise method definitions, guidelines, templates, principles, best practices, internal procedures and regulations, training material and any other general descriptions of how they want and need to develop software.

The information can be used for reference and education and as the basis for developing standard processes. All the managed content can then be published to html and deployed to servers for distributed use.

Typically, a Company needs to address two key problems in order to deploy new processes successfully:

1. development teams need to be educated on the methods applicable to the roles for which they are responsible;
2. development teams need to understand how to apply these methods throughout the development lifecycle. That means they need to define or select a process and they need a clear understanding of how the different tasks relate to each other;

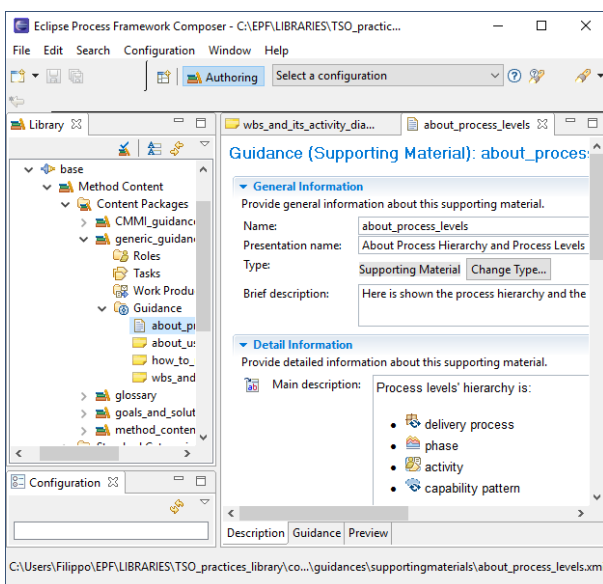


Figure C.1 – Exemplary tool: EPF Composer

For these purposes, the exemplary tool EPF Composer provides to the user the following capabilities:

Method Authoring

Best practices can be captured as a set of reusable method building blocks as defined in the meta-model; roles, work products, tasks, and guidance, such as templates, guidelines, examples, and check lists. A rich-text editor allows you to document method elements, and graphical views present diagrams showing relevant relationships.

Reuse is facilitated by allowing you to create a method element as a derivative of another method element through various inheritance-type of relationships. This allows you to e.g. specify that a Systems Architect has similar responsibilities to a Software Architect by expressing the differences, reusing everything that is common.

Process Authoring

Reusable process building blocks can be organized into processes along a lifecycle dimension by defining e.g. Work Breakdown Structures (WBSs), and when in the lifecycle to produce what work products in which state.

The tool allows you to construct reusable chunks of processes through so called capability patterns. A capability pattern may for example explain how to define, design, implement and test a scenario or a user story, and this pattern can now be reused in a variety of processes. The tool also allows you to define delivery processes, which are end-to-end processes. Structural information can often be edited with graphical as well as non-graphical editors.

Library Management and Content Extensibility

An XMI-based library enables persistency and flexible configuration management as well as content interchange for distributed client-server implementations. Method and process content can be packaged into content plug-ins and content packages allowing simple distribution, management and extensibility of content. As content plug-ins are added to your content library, the tool will resolve dependencies between process elements.

Configuring and Publishing

A process configuration can be created by selecting a set of content plug-ins and content packages.

Optionally, an exemplary process configuration can be used as a starting point, and content plug-ins and content packages added or removed from this exemplary configuration.

As an example, you may start with a generic exemplary process suitable for small collocated teams and add content plug-ins containing specific guidance for each of Eclipse, JUnit, J2EE, and IBM Rational RequisitePro. The delivery processes associated with a configuration can be further customized.

As the configuration is published, the tool resolves the many relationships that may exist between process elements in the selected plug-ins and packages, and generates a set of html pages with links representing relationships between process elements to make the resulting Web site easy to navigate. The resulting Web site is viewable via a web browser, without the need for a Web server. This will allow users on all platforms to view the published process.

C.3 PLUG-INS LOGICS

All the method library content is organized in modular units called method plug-ins.

A new method plug-in:

- starts out as a standalone element, with no linking to any other plug-in;
- has a designated addressing defined by its complete name;
- has no content inside, but presents a default setting to insert method content, capability patterns and delivery processes (also see *Figure C.2*);
- may be linked to other plug-ins by ticking their names in the 'Referenced Plug-ins' box, so permitting the new plug-in to use their method content and processes;
- may be filled with new method content and processes;
- can be exported from or imported to any EPF Composer method library (paying attention, though, to the active references).

References are a very critical feature of EPF Composer. They make possible to effectively recall and reuse existing content but, on the other hand, an inadequate management can cause sever-

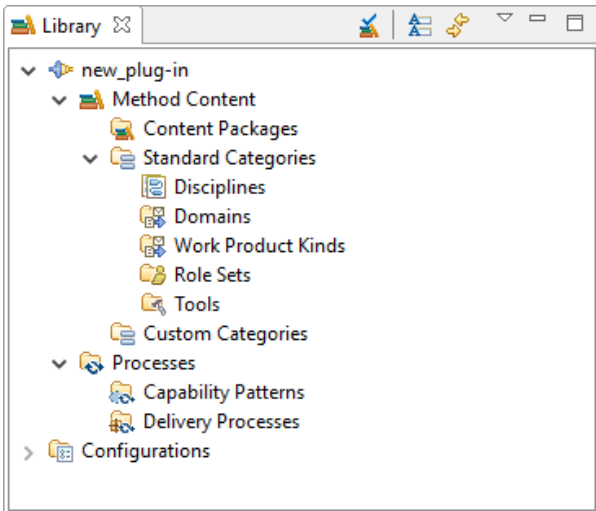


Figure C.2 – Setting of a new plug-in

al problems for what concerns reusing content, publishing it, exporting/importing plug-ins. Content developers should be aware of the possible problems and pay a particular attention at avoiding circular references.

In fact, references must present a tree-structure consisting of many different chains of references, in which a plug-in that lays at the top of a chain can draw every content of the plug-ins it finds along that chain.

C.4 EPF METHOD FRAMEWORK

The most fundamental principle of EPF Composer is the separation of reusable core method content

from its application in processes. Almost all the EPF Composer concepts are categorised along this separation.

Method content describes what is to be produced; the necessary skills required and the step-by-step explanations describing how specific development goals are achieved. These method content descriptions are independent of a development lifecycle.

Processes describe the development lifecycle, taking the method content elements and relating them into semi-ordered sequences that are customised to specific types of projects.

Figure C.3 provides a summary of the key elements used in EPF Composer and their relationships with processes and/or method content.

As it can be observed, method content is primarily expressed using work products, roles, tasks, and guidance. Guidance, such as checklists, examples, or roadmaps, can also be defined to provide exemplary walkthroughs of a process.

On the right-hand side of the diagram there are the elements used to represent processes in EPF Composer. The main process element is the activity, that can be nested to define a work breakdown structure and be related with other process elements to define a flow of work. Activities also contain descriptors base on method content elements. The two main types of process supported by EPF Composer are the delivery process and the capability pattern; they both can be built of activities.

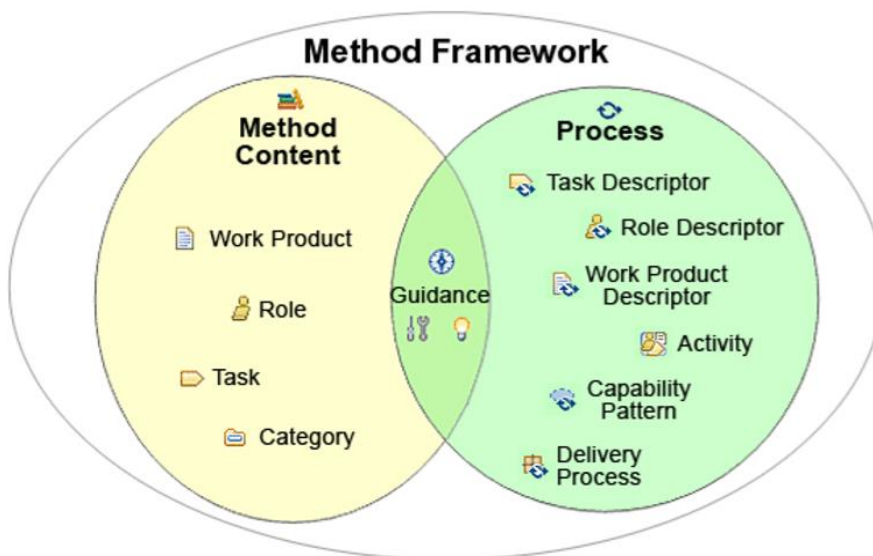


Figure C.3 – EPF method framework

C.5 EPF COMPOSER ELEMENTS

All the method content and process elements introduced in the previous chapter are hereafter listed and described.

Task

A task defines a unit of work that needs to be done to transform inputs into outputs through a series of steps performed by one or more roles independent of a particular work breakdown structure (WBS). A task may be divided into steps.

Role

It describes a standard set of responsibilities and corresponding skills necessary to perform a task or create a work product. A role does not correspond with a single person. In fact, a same person may execute several roles simultaneously or during the course of a project, and a role may likewise be defined to represent a group such as a review board.

Work Product

It is used to define and describe the items needed as input or created as output of one or more tasks that are the responsibility of a role. EPF Composer allows three types of work products: artefacts, outcomes and deliverables.

An artefact is a tangible work product that is consumed, produced, or modified by one or more tasks. Artefacts may be composed of other artefacts. An outcome is an intangible work product that may be a result or state. It may also be used to describe work products that are not formally defined. A deliverable is a collection of work products, usually artefacts, used to define typical or recommended content in the form of work products packaged for delivery.

Guidance

A General term referring to all types of material that provide additional detail on other types of elements:

- Checklist: it identifies a series of items that need to be completed or verified. Checklists are often used in reviews such as walkthroughs or inspections.
- Concept: it outlines key ideas or basic principles that serve as foundation for additional information.

- Example: used to include typical samples of the items to be produced, may often only be a partial sample that is intended as further guidance rather than something to be reused.
- Guideline: it provides additional detail on how to handle a particular method element. Guidelines most commonly describe how to perform some set of actions related to tasks or provide additional rules or recommendations related to the representation of work products.
- Estimation consideration: it describes the amount of effort to produce a work product or perform a task including any influencing factors.
- Practice: it describes a proven way of doing something or common approaches and strategies that represent best practices. This is also used to represent standards and policies related to methods.
- Report: used to provide guidance on representing the output of an automated tool that may be a combination of information from one or more other work products.
- Reusable asset: linking to intellectual capital that can be utilized to perform some task or leveraged as a starting point for the creation of a solution. This type of guidance is usually represented as a link to some external source. This may include assets such as source code, templates, patterns, architectural frameworks, domain models, and so on – that can be reused in a different context.
- Roadmap: specific to a process that represents a linear walkthrough of those items from a particular perspective.
- Supporting material: catch-all for other types of guidance not specifically defined elsewhere.
- Template: it specifies the structure of a work product by providing a pre-defined table of contents, sections, packages, and/or headings, a standardized format, as well as descriptions on how the sections and packages are supposed to be used and completed. Often provided as a form or empty instanced of a work product that can be used as starting point for the creation of a new one.
- Term definition: provides definitions that are used to build up the library glossary.

- Tool mentor: explains how to apply a specific tool to accomplish a task, perform a set of steps or instantiate a particular work product.
- White paper: it represents externally published papers that can be read and understood in isolation of other content elements.

Custom Category

Used to categorize content based on the user's criteria. One important use is, as already stated, for constructing views for publishing.

Activity

In the UMA an activity is a breakdown element which supports the nesting and logical grouping of related process elements, such as descriptor and sub-activities, thus forming the breakdown structures.

Like any other process or sub-process, the activity can generate an activity diagram to graphically show the relationships between the work breakdown elements (WBEs).

Capability Pattern

A sub-process that expresses and communicates process knowledge for a key area of interest, such as a discipline or a best practice. Capability patterns are also used as building blocks to assemble delivery processes or larger capability patterns. This ensures optimal reuse and application of their key best practices in process authoring activities in EPF Composer.

Delivery Process

It represents a complete and integrated process template for performing one specific type of project. It describes a complete end-to-end project lifecycle and it is used as a reference for running projects with similar characteristics.

Descriptor

It defines how method content is represented in a process. Like shown in figure, there are three kinds of descriptors: task descriptor, role descriptor and work product descriptor.

They are the key concept for realizing the separation of process from method content. A descriptor has its own relationships and properties which can be modified rather independently of the default relationships defined in the method content.

REFERENCES AND RESOURCES LIST

- Systems Engineering Fundamentals, Defense Acquisition University Press, January 2001
https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-885j-aircraft-systems-engineering-fall-2005/readings/sefguide_01_01.pdf
- CMMI Institute
<https://cmmiinstitute.com>
- CMMI for Development v. 1.3, CMMI Product Team, November 2010
https://resources.sei.cmu.edu/asset_files/TechnicalReport/2010_005_001_15287.pdf
- CMMI for Development v. 1.3 – WIT (Web Idea Tree) version
<http://cmmis.free.fr/cmmi-dev/text/pa-rm.php>
- CMMI Model Foundation – What is it?
<http://www.plays-in-business.com/cmmi-model-foundation-what-is-it/>
- Eclipse Process Framework Project
<https://projects.eclipse.org/projects/technology.epf>
- Eclipse Process Framework (EPF) Composer - Installation, Introduction, Tutorial and Manual, Bjorn Tuft, March 2010
https://www.eclipse.org/epf/general/EPF_Installation_Tutorial_User_Manual.pdf
- Increasing Development Knowledge with EPFC, Peter Haumer, 2006
<http://www.haumer.net/paper/EPFC-eclipsereview.pdf>
- Microsoft Project
<https://products.office.com/en-us/Project/project-and-portfolio-management-software>
- EPF Composer
<http://www.eclipse.org/downloads/download.php?file=/technology/epf/composer/release/epf-composer-1.5.2-win32.zip>
- EPF Practices 1.5.1.5
http://www.eclipse.org/downloads/download.php?file=/technology/epf/PracticesLibrary/library/epf_practices_library_1.5.1.5_20121212.zip

ACRONYMS

CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
CMMI-DEV	Capability Maturity Model Integration for DEvelopment
ECS	Environmental Control System
EMF	Eclipse Modeling Framework
EPF	Eclipse Process Framework
HTE	How To Expand
HTN	How To Navigate
HWD	How Was Developed
LAD	Leonardo Aircraft Division
OMG	Object Management Group
OPAL	Organizational Process Asset Library
TSO	Training Systems Organization
TPL	TSO Practices Library
TYP	Tailor Your Process
SEP	Standard Enterprise Practices
SPEM	Software Process Engineering Meta-model
UAV	Unmanned Aerial Vehicle
UMA	Unified Method Architecture
WBE	Work Breakdown Element
WBS	Work Breakdown Structure
XMI	XMI Metadata Interchange
XML	eXtensible Markup Language

