

# Implementation of a Software Process Standard as an Electronic Process Guide

Simon Alexandre

*CETIC - Centre d'Excellence en Technologies  
de l'Information et de la Communication*

[simon.alexandre@cetic.be](mailto:simon.alexandre@cetic.be)

Timo Mäkinen & Timo Varkoi

*Tampere University of Technology, Pori*  
[timo.makinen@tut.fi](mailto:timo.makinen@tut.fi), [timo.varkoi@tut.fi](mailto:timo.varkoi@tut.fi)

## Abstract

*International standards for very small software enterprises are emerging. Issues related to dissemination of the standards include usability and affordability for small organizations. An attractive way to present the standards would probably gain new users. The presentation platform for the standards should be based on tools that are easily available. This paper describes an example of an implementation of life cycle profiles for very small enterprises using Eclipse Process Framework Composer. The mapping of the process models is presented and implementation issues are discussed.*

## 1. Introduction

International Organization for Standardization (ISO) publishes a number of standards also for software and system engineering domain. Typically these standards are multi-part paper documents. Novice users may find it hard to find the right set of standards for their purposes. Especially small and medium-sized enterprises (SME) do not exploit the standards as much as they could to successfully compete in the market.

As the importance of SME is acknowledged there are also a growing number of efforts to produce international standards for them. This forces the standard developers to think of new, more attractive ways to offer the standards.

An example of software engineering standards for SME is the newly established work to provide small companies with Software life cycle profiles and guidelines for very small enterprises (VSE). One of the aims of the responsible standardization body is to produce guides that are “understandable, affordable and usable by VSE” as stated in working group’s internal requirements document.

As a result of this study we represent an alternative to disseminate a process oriented standard using affordable, easy to use tools. We have used these tools in similar contexts in our earlier research: already implemented process libraries include browsers for ISO/IEC 15504-5 An exemplar Process Assessment Model for software processes (SPICE) and IT Infrastructure Library processes for service management. We also participate in

the development of the VSE standard set. The presented implementation will also serve as a reusable process library for process assessment and improvement.

This paper gives elementary information about the organization and effort to create a software process standard for VSE and describes the basic elements of the emerging standard in chapter two. An applicable tool, Eclipse Process Framework (EPF) Composer, is presented and issues related to its process model are discussed in chapter three. Chapter four presents a mapping between the VSE basic elements and EPF, and discusses the implementation issues. Chapter five summarizes the paper and discusses future ideas. Examples of the EPF implementation are in the Appendix one.

## 2. Life Cycle Processes for Very Small Software Enterprises

### 2.1 Emerging international standard

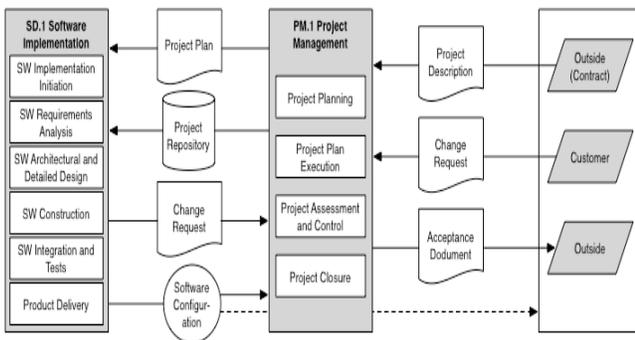
Even the smallest software enterprises have increased their significance to the world-wide industry and this makes them also a target for the international standardization efforts. One of the important parties in providing international standards is the joint technical committee of the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC), named Joint Technical Committee One (ISO/IEC JTC 1). JTC 1 subcommittee seven (SC7) has set up a working group (WG) to develop standards for software lifecycle profiles to help very small enterprises (VSE) to enhance quality of their processes and products.

WG 24 - Software life cycle profiles and guidelines for very small enterprises - was established in 2005. In this context a very small enterprise is defined to be a company or an organization with less than 25 employees. The working group prepares a set of work products, technical reports (TR) and international standardized profiles (ISP), under the generic title Lifecycle Profiles for Very Small Enterprises (VSEP):

- TR 29110-1 Overview

- ISP 29110-2 Framework and Taxonomy
- TR 29110-3 Profile Assessment Guide
- ISP 29110-4.1 Basic Profile Specification
- TR 29110-5.1 Management and Engineering Guide for Basic Profile

WG 24 has recently published the first drafts of the 29110 standard. Considering deployment of the standard, part 5.1 [6] is the most interesting as it defines in practice a software implementation and management guide appropriate to VSE. The Basic Profile is composed of two processes: Project Management and Software Implementation. WG 24 plans to develop supplementary profiles to cover more of the life cycle processes in order to support VSE to achieve adequate process maturity for ISO 9001 certification.



**Figure 1. Context of VSEP Basic Profile Processes with their subprocesses. [7]**

Figure 1 depicts the VSEP Basic Profile Processes in their context. There are two processes: *Software Implementation* and *Project Management*. Software Implementation is divided into six subprocesses and Project Management into four subprocesses.

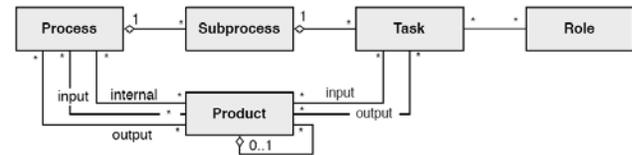
Project Management acts as the interface to the *outside* world. The opening document for a software project is *Project Description*, which is also the basis for project planning. Project Management generates a *Project Plan* to direct the software project and establishes a *Project Repository* to store project work products. During the execution of the project, Project Management process receives *Change Requests*, which might cause revisions to the Project Plan. The source of a Change Request is either one of the processes or *Customer*. The final outcome of the Software Implementation process is a *Software Configuration*, which includes, in addition to executable software and its source, all associated documentation. The customer acceptance is formalized by *Acceptance Document*. [7]

## 2.2 VSEP Basic Profile process meta-model

In the VSEP Basic Profile, the following main elements are used to describe processes: Process,

Subprocess, Task, Product and Role (Figure 2). Process, Subprocess and Task represent a hierarchy of activities. All the instances of Process and Subprocess are depicted in Figure 1. As an example, the tasks of Software Design subprocess are shown in the Figure A2 (in Appendix).

Between Process and Product there are three associations. A product is acting as an input, output or internal product of a process. For example (Figure 1) Project Plan and Acceptance Documents are output products of Project Management Process, and one of its inputs is Project Description. An example of Project Management process's internal products is Progress Status Record. Product may have a structure like Project Plan is included in Project Repository.



**Figure 2. Elements of a process in VSEP Basic Profile.**

Task is related to Product with two associations: input and output, and between Task and Role there is one association. For instance (see Figure A3), Software Design subprocess contains the task Document Design, of which input is Requirements Specification while its outputs are Software Design and Traceability Record. The performing roles of the Document Design task are Designer, Analyst and User Interface Designer.

An additional element in VSEP Basic Profile process is Resource, which is related to Subprocess. However, resources are expressed in the TR in quite a general fashion (like “Construction tools” as the resource of Software Construction subprocess), and that is why Resource is omitted from Figure 2. A more detailed analysis of the VSEP Basic Profile's process meta-model is shown in [7]. There is also a short discussion, how to improve the model's traceability, balance and consistency.

## 3. Eclipse Process Framework

### 3.1 History and key concepts

Eclipse is a free and open source Integrated Development Environment (IDE) originally initiated by IBM in 2001 [1]. Three years later, the Eclipse foundation was created as an independent not-for-profit corporation to act as the steward of the Eclipse community. Today, the Eclipse open source community is made of individuals and organizations from a cross section of the whole software industry. The Eclipse platform is used by

thousands of IT companies developing software. This success is mainly due to the design of the Eclipse platform offering a documented API and environment facilitating the development and integration of plug-ins. Currently, hundreds of plug-ins (e.g. unit test, UML modeling, and version control) can be used with Eclipse.

The Eclipse Process Framework Composer (EPF Composer) project started in February 2006 and published its first release 1.0 in September 2006 [2]. EPF Composer is a process management tool platform and conceptual framework for authoring, tailoring and deploying software development processes. EPF Composer has been developed to address process related problems faced by project or program managers and by process engineers in charge of defining and maintaining development processes. Two major problems are addressed by the tool. Firstly, it facilitates the understanding of software development methods, processes and concepts by project teams. Secondly, it explains to development team how to apply a given methodology by describing the sequence of steps, explaining the techniques and tools to be used to perform a given task [4].

The key concepts handled by EPF Composer, Methods and Processes are coming from the Unified Process (UP). Method content describes what is to be produced, necessary skills and detailed description of steps to be performed to achieve development goals. Method content comes from books, technical papers, standards, best practices or homegrown methods. Processes define the sequence of steps, the roles and the work products that are used and produced over time to achieve a given purpose. Processes describe the development lifecycle and take method elements that are related into a sequence customized for projects. EPF Composer does not imply any particular development approach by allowing selection of any method content and open design of processes.

Method content is expressed by five concepts or elements: Work Product, Role, Task, Category and Guidance (common to Method content and Process). Processes in EPF composer are defined using six concepts. The main concept is the Activity that can either take the shape of a Capability Pattern (processes expressing process knowledge in a key area such as a discipline or a best practice) or a Delivery Process (complete and integrated process template to achieve a specific type of project). Additionally, Task Descriptor, Role Descriptor and Work Product Descriptor concepts are used in processes description [4].

The theoretical EPF Composer authoring approach [5] can be divided into four steps. The first step consists of defining the method content by gathering, analyzing and extracting information related to a development methodology (e.g. an agile development method). In the

second step, the set of necessary processes are identified and defined (e.g. processes for embedded software development). In the third step, the comprehensive and integrated process framework is configured to fit project needs. The fourth and last step consists of creating project plan templates to enact processes in the context of a specific project. However, other approaches can be followed as the tailoring, to fit specific needs and context, of available processes that can be downloaded from the Eclipse Process Framework web site.

Currently, EPF project offers the EPC Composer editor version 1.2.0.2 and three exemplar process models that are Open UP, Scrum and Extreme Programming.

### 3.2 EPF Process Meta-model

This section takes a brief look to the EPF process meta-model. The focus is on the elements, which are used to store the VSEP Basic Profile in EPF process library. The EPF elements are classified into two main groups: Method Content and Process. Elements from both of the groups are used in the implementation of the VSEP Basic Profile as content for EPF.

“EPF **method content** describes roles, the tasks that they perform, the work products produced by the tasks performed and supporting guidance. They can be categorized into logical groups for indexing and display purposes. Method content elements are independent of a development lifecycle. In fact, they are often reused in multiple development lifecycles.” [3]

Figure 3 depicts a subset of the EPF method content elements. The gray-shaded classes arguably seem to be counterparts to the VSEP process elements: Subprocess (VSEP) - Discipline (EPF), Task - Task, Role - Role, and Product - Work Product. However the associations between the elements are different.

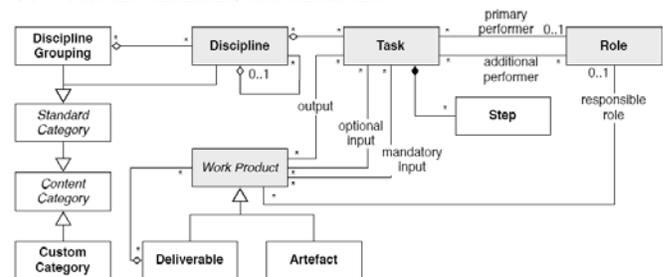


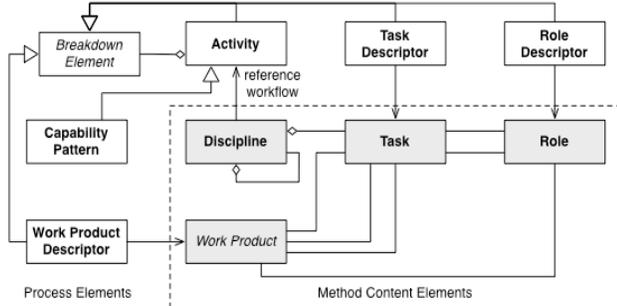
Figure 3. EPF method content elements.

In VSEP, between Task and Role, there is only one association, when in EPF there is two. EPF’s Role acts as Task’s primary or additional performer. EPF also divides Task into Steps, which are not clearly identifiable in VSEP. EPF’s Discipline groups Tasks, which are grouped by Subprocess in VSEP. In both of the models, Product (or Work Product) is Task’s output, but unlike VSEP, EPF divides input associations into mandatory and

optional. VSEP's Product has a structure. In EPF, Work Product has subclasses, of which only Deliverable has a structure.

The counterpart for VSEP's Process is not so clear. EPF's Discipline can be grouped by Discipline Grouping or by another Discipline. In VSEP, Process groups Subprocesses, but between Process and Product there are associations, which do not exist in EPF between Discipline/Discipline Grouping and Work Product. A conceivable solution for the problem is Custom Category, which can associate with any element.

"EPF processes describe the development lifecycle. They define sequences of tasks performed by roles, and work products produced over time. Processes are typically expressed as workflows or breakdown structures. The sequencing of the tasks within the breakdown structure usually represents different types of development lifecycles, such as waterfall, incremental, and iterative." [3]



**Figure 4. EPF process elements and their relationships with method content elements.**

Figure 4 shows a subset of the EPF process elements and their relationships to the (gray-shaded) content elements. In processes, Descriptors refer to the corresponding content elements (Task, Work Product, Role) for reusing them. The descriptors are special types of Breakdown Element, which are grouped by Activity. For Activity, there are three subclasses: Process, Phase and Iteration. Process is either (a "complete") Delivery Process or Capability Pattern, which can be reused in several Activities (e.g. Phases) of a Delivery Process.

The VSEP Basic Profile processes are also shown as workflows of subprocesses. Because it is not possible to describe a workflow by method content elements, the VSEP's EPF implementation requires EPF process elements, too. A promising counterpart for VSEP Process seems to be Capability Pattern, because, like VSEP Process, it is a top level process element in pursuance of describing a reusable part of a software process. As discussed above, Discipline can be considered as the EPF content equivalent for the VSEP subprocess. A Discipline is also able to refer Activities of the EPF process. Thus an Activity, which is a part of the Capability Pattern, may be the counterpart for a VSEP subprocess.

## 4. VSEP Basic Profile as EPF Content

### 4.1 Process Meta-Model Mapping

The EPF counterparts to the VSEP process elements are summarized in Table 1, and a set of mappings of the associations is shown in Table 2. Only the associations, which are significantly different among VSEP and EPF, are represented.

The three associations between Process and Product (input, output, internal) are described by Custom Categories. All of the tasks' inputs are regarded mandatory. The first mentioned task performer is considered to be primary, and the others are optional.

**Table 1. Mapping of the VSEP process elements to EPF.**

VSEP	EPF (Method)	EPF (Process)
Process	Custom Category	Process Pattern
Subprocess	Discipline	Activity
Task	Task, Step	Task Descriptor
Product	Deliverable, Artifact	Work Product Descriptor
Role	Role	Role Descriptor

**Table 2. Mapping of VSEP associations to EPF.**

VSEP	EPF
Process input Process output Process internal (product)	Custom Category
Task input (product)	Task mandatory input
	Task optional input
Task performer (role)	Task primary performer Task additional performer

### 4.2 Implementation of a Process Library

Figure 5 depicts the tree frame of the www-site, which has been generated from the VSEP Basic Profile EPF library. The content frames of the highlighted items are shown in Appendix 1: Software Implementation Process (EPF Custom Category), Software Design Subprocess

(EPF Discipline), Document Design Task (EPF Task), and Software Design Subprocess (EPF Activity).

In the VSEP Basic Profile tree, there are two branches; one for the method content elements (Method Content) and the other for the process elements (Process Patterns). The basic process element classes constitute the second level in the Method Content branch: Processes and Tasks, Work Products, and Roles, of which the first one is opened to show the different levels in the VSEP activity hierarchy. Custom Category, which is denoted in Figure 5 by a folder icon, was applied to construct the upper levels of the branches.

In the Method Content branch the VSEP processes are directly under the Processes and Tasks folder, and the associations of a process are on the following level. As shown in Figure 5 for Software Implementation process, Custom Categories are also used to describe a process and its links to products and subprocesses. The disciplines and furthermore the tasks can be found under the Subprocesses folder. In addition to the tasks, there are activities under a discipline. For example, Software Design activity is a part of Software Implementation capability pattern, but in the tree, it is also under Software Design discipline, because the activity is the discipline's reference workflow.

The Process Pattern branch includes only the activity hierarchy: Processes (Capability Patterns), Subprocesses (Activities) and Tasks (Task Descriptors). The other process elements, Role Descriptors and Work Product Descriptors, can be reached by the links in the content frames.

Figure A1 depicts the content frame of Software Implementation Process as EPF Custom Category. The frame includes three attributes and links to the associated frames. The Custom Category's Presentation Name, Brief Description and Main Description are used for the Process's Title, Purpose and Objectives respectively. The links corresponds to the position of the process in the tree. The link with the label Categories points upwards and Contents links downwards.

The frame for Custom Category is very similar to Discipline's frame, of which example is shown in the Figure A2. However, there are no Content links, but links to the frames of tasks and the activity, which is acting as the reference workflow of the activity. There is also a label ("Discipline") before the discipline's Presentation name that does not exist in the case of Custom Category.

In Figure A3, there is an example of Task's content frame. There are some differences comparing it to the content frame of Discipline or Custom Category. Firstly, there is a structured and multi-valued attribute called Step, which includes a name and a description. In the example, there are four steps. One step's description is shown. Secondly, there are links to the frames of many types of elements. The links corresponds to the Task's

associations, which are shown in Figures 3 and 4. The link to the Discipline's content frame is just under the Task's Brief Description while all the other links are in the Relationship section. The Process Usage links points, in addition to the corresponding Task Descriptor, to the Activity, which includes the Task Descriptor, and to the Capability Pattern, which includes the Activity.



**Figure 5. The tree frame of the VSEP Basic Profile generated from the EPF process library.**

VSEP Task has only Number and Description as attributes, which have been transformed to the following EPF Task attributes: Presentation Name, Brief Description and Steps with Name and Description. The Brief Description is considered to be the first sentence of the VSEP Task Description. If there are other sentences, they have become the Step Descriptions. The name for a Step is a predicate with an object, which have been

selected from the Step's description. The Task's Presentation Name has been build respectively, but it includes also the Number of the VSEP Task.

Figure A4 depicts a part of the content frame of Software Design subprocess as Activity. The frame includes four tabs for different pages of information. The Work Breakdown Structure tab has been selected. The page shows, in addition to the list of the work items on the next level, a set of diagrams. The figure illustrates the Activity Diagram and a part of the Activity Detail Diagram of Software Design subprocess.

## 5. Conclusions and Future Work

We presented a brief overview of a standardization effort that aims in creating Software life cycle profiles and guidelines for very small enterprises. Usability and affordability are issues that might affect the popularity of the standard among the VSE. As a solution we describe EPF Composer based process library. The content of this library can be generated into a www-site.

Based on the output of this work, one of the next tasks to answer VSE particular context and needs would be to include in the current process framework some exemplar lifecycle elements coming from development methodologies. Among the candidates we identify the Unified Process (i.e. OpenUP), already provided with EPF. This development methodology could be tailored to fit the requirements of the first VSE profile defined by WG24.

In the future, the reusable process library will also serve as a basis for process assessment and improvement methods for VSE. The results of this study support the future standardization work by providing a concrete starting point for evaluation and development of the possible dissemination solution for the merging standard.

The next steps of the VSEP standard development will extend the profiles with not only additional processes but also with process capability attributes. This offers new challenges also for the implementation of the process library.

## 6. References

- [1] Eclipse, <http://www.eclipse.org> . 2008.
- [2] Eclipse Process Framework, <http://www.eclipse.org/epf/> . 2008.
- [3] Eclipse Process Framework (EPF) Composer 1.0 Architecture Overview, [http://www.eclipse.org/epf/composer\\_architecture/](http://www.eclipse.org/epf/composer_architecture/) . 2008.
- [4] Haumer, P., Eclipse Process Framework Composer. Part 1: Key Concepts Second Revision. 2007.

[5] Haumer, P., Eclipse Process Framework Composer. Part 2: Authoring method content and processes. 2007.

[6] ISO/IEC PDTR 29110-5.1.WD 2, Software Engineering — Lifecycle Profiles for Very Small Enterprises (VSE) — Part 5.1: Management and Engineering Guide - Basic Profile. 2007.

[7] Mäkinen, T. & Varkoi, T. (2008). Analyzing a Process Profile for Very Small Software Enterprises. Accepted to SPICE 2008 Conference. Nuremberg Germany, May 26-28 2008.

## Appendix 1.

- A.1 Software Implementation Process (EPF Custom Category)
- A.2 Software Design Subprocess (EPF Discipline)
- A.3 Document Design Task (EPF Task)
- A.4 Software Design Subprocess (EPF Activity)

## A.1 Software Implementation Process (EPF Custom Category) - Fig. A1

### Software Implementation (SD.1)



The purpose of the Software Implementation process is the systematic performance of the analysis, design, construction, integration and tests subprocesses for new or modified software products according to the specified requirements.

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

Relationships	
Categories	<ul style="list-style-type: none"><li>Processes and Tasks</li></ul>
Contents	<ul style="list-style-type: none"><li>SD Subprocesses</li><li>SD Input Products</li><li>SD Output Products</li><li>SD Internal Products</li></ul>

[Back to top](#)

Main Description
<p><b>Objectives</b></p> <p>01. The tasks of the subprocesses are performed through the accomplishment of the current Project Plan.</p> <p>02. Software requirements are defined, analyzed for correctness and testability, approved by the Customer, baselined, communicated, and changes to them are evaluated for cost, schedule and technical impact, previously to be processed.</p> <p>03. A software architectural and detailed design is developed and baselined, it describes the software items; internal and external interfaces of them, consistency and traceability to software requirements are established.</p> <p>04. Software components defined by the design are produced; unit test are defined and performed to verify the consistency with requirements and the design, traceability to requirements and design are established.</p> <p>05. Software configuration is produced performing integration of software components and verified using Test Cases and Test Procedures, results are recorded at the Test Incident Report, defects are corrected and consistency and traceability to Software Design are established.</p>

## A.2 Software Design Subprocess (EPF Discipline) - Fig. A2

### Discipline: Software Design



The software architecture and detailed design subprocess transforms the software requirements to the system software architecture and software detailed design.

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

Relationships	
Categories	<ul style="list-style-type: none"><li>SD Subprocesses</li></ul>
Reference Workflows	<ul style="list-style-type: none"><li>Software Design</li></ul>
Tasks	<ul style="list-style-type: none"><li>SD 3.01 Assign Tasks</li><li>SD 3.02 Understand Requirements</li><li>SD 3.03 Document Design</li><li>SD 3.04 Verify Design</li><li>SD 3.05 Correct Design (ver)</li><li>SD 3.06 Incorporate to Configuration</li></ul>

[Back to top](#)

Main Description
<p>The subprocess provides:</p> <ul style="list-style-type: none"><li>Work team review of the requirements specification to identify the top-level software structure, the software components and associated interfaces</li><li>Allocated software requirements to the top-level structure</li><li>Defined architecture alternatives and the software architecture solution</li><li>Detailed design of the components and interfaces</li><li>Software design verified against the design criteria and corrected defects</li><li>Traceability of the software requirements to the software components</li><li>Design products and documents under version control</li></ul>

### A.3 Document Design Task (EPF Task) - Fig. A3

#### Task: SD 3.03 Document Design

 Document or update the Software Design.  
Disciplines: Software Design

 Expand All Sections  Collapse All Sections

Relationships		
<b>Roles</b>	<b>Primary Performer:</b> <ul style="list-style-type: none"> <li>Designer (DES)</li> </ul>	<b>Additional Performers:</b> <ul style="list-style-type: none"> <li>Analyst (AN)</li> <li>User Interface Designer (UID)</li> </ul>
<b>Inputs</b>	<b>Mandatory:</b> <ul style="list-style-type: none"> <li>Requirements Specification (6)</li> </ul>	<b>Optional:</b> <ul style="list-style-type: none"> <li>None</li> </ul>
<b>Outputs</b>	<ul style="list-style-type: none"> <li>Software Design (7)</li> <li>Traceability Record (8)</li> </ul>	
<b>Process Usage</b>	<ul style="list-style-type: none"> <li>Software Implementation &gt; Software Design &gt; Document Design</li> </ul>	

 Back to top

**Steps**

 Expand All Steps  Collapse All Steps

 **Generate Architectural Design**

Analyze the Requirements Specification to generate the architectural design, its arrangement in subsystems and components defining the internal and external interfaces.

---

 Describe Interfaces

 Describe Components

 Generate Traceability Record

### A.4 Software Design Subprocess (EPF Activity) - Fig. A4

#### Activity: Software Design

 The software architecture and detailed design subprocess transforms the software requirements to the system software architecture and software detailed design.

**Description** | **Work Breakdown Structure** | **Team Allocation** | **Work Product Usage**

