**Deployment Package**

**Functional & Physical Architecture (FA & PA)**

**Systems Engineering Basic Profile**

**Notes:**

This document is the intellectual propriety of its author’s organization. However, information contained in this document is free to use. The distribution of all or parts of this document is authorized for non-commercial use as long as the following legal notice is mentioned:

© International Council on Systems Engineering (INCOSE) 2013

Commercial use of this document is strictly forbidden. This document is distributed in order to enhance exchange of technical and scientific information.

This material is furnished on an “as-is” basis. The author(s) make(s) no warranties of any kind, either expressed or implied, as to any matter including, but not limited to, warranty of fitness for purpose or merchantability, exclusivity, or results obtained from use of the material.

The processes described in this Deployment Package are not intended to preclude or discourage the use of additional processes that Very Small Entities may find useful.

|  |  |
| --- | --- |
| **Authors** | Sascha Ackva – INCOSE, Germany |
| **Editors** | Joseph W. Marvin - INCOSE, USA  Claude Y Laporte – École de technologie supérieure (ETS), Canada  Martin GEISREITER – INCOSE, Germany  *To be completed* |
| **Creation date** | 09 April 2013 (*Day-Month-year (e.g. 12 June 2012))* |
| **Last update** | 2013-04-09 *(Day-Month-year)* |
| **Version** | 1.0 *(X.X)* |

Version History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date**  (yyyy-mm-dd) | **Version** | **Description** | **Author** |
| 2013-04-09 | 0.1 | Initial Release | Sascha Ackva |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Abbreviations/Acronyms

|  |  |
| --- | --- |
| **Abre./Acro.** | **Definitions** |
| DP | Deployment Package - a set of artifacts developed to facilitate the implementation of a set of practices, of the selected framework, in a Very Small Entity. |
| VSE | Very Small Entity – an enterprise, organization, department or project having up to 25 people. |
| VSEs | Very Small Entities |
| ISO | International Organization for Standardization, <http://www.iso.org> |
| INCOSE | International Council on Systems Engineering, <http://www.incose.org/> |
| V&V | Verification and Validation |
| PMBOK | Project Management Body of Knowledge, <http://www.pmi.org/> |
| *<details>* | *<details>* |
|  |  |

Table of Contents

[1 Technical Description 5](#_Toc355197504)

[Purpose of this document 5](#_Toc355197505)

[Why is Systems Engineering important? 5](#_Toc355197506)

[Why is cooperation between Systems Engineering and Project Management important? 6](#_Toc355197507)

[Why is Functional Analysis and Physical Architecture Important? 7](#_Toc355197508)

[Tailoring this Deployment Package 9](#_Toc355197509)

[2 Definitions 10](#_Toc355197510)

[Generic Terms 10](#_Toc355197511)

[Specific Terms 10](#_Toc355197512)

[3 Relationships with ISO/IEC 29110 12](#_Toc355197513)

[4 Description of Processes, Activities, Tasks, Steps and Products 15](#_Toc355197514)

[Plan the systems engineering approach 15](#_Toc355197515)

[Identify system functional input requirements and needs 17](#_Toc355197516)

[Identify System Boundary and Use Cases 19](#_Toc355197517)

[Describe use case with operational scenarios and related interfaces 20](#_Toc355197518)

[Identify Logical Architecture 22](#_Toc355197519)

[Allocate functions to logical architecture 23](#_Toc355197520)

[Identify Physical Architecture 25](#_Toc355197521)

[Role Description 28](#_Toc355197522)

[Product & Artifacts Descriptions 29](#_Toc355197523)

[5 Templates 33](#_Toc355197524)

[Template: Systems Engineering Management Plan (SEMP) 33](#_Toc355197525)

[Template: System Requirements Specification 34](#_Toc355197526)

[Template: System Design Document 34](#_Toc355197527)

[6 Example 35](#_Toc355197528)

[7 Checklist 36](#_Toc355197529)

[8 Tools 37](#_Toc355197530)

[9 References to Other Standards and Models 38](#_Toc355197531)

[ISO/IEC 15288 Coverage Matrix 38](#_Toc355197532)

[CMMI for Development, Version 1.3 Coverage Matrix 38](#_Toc355197533)

[ISO 21500 Coverage Matrix 39](#_Toc355197534)

[10 References 40](#_Toc355197535)

[11 Evaluation Form 41](#_Toc355197536)

# Technical Description

## Purpose of this document

A Deployment Package (DP) is a set of artifacts developed to facilitate the implementation of a set of practices in a Very Small Entity (VSE). A DP is not a process reference model (i.e. it is not prescriptive). The elements of a typical DP are: roles and products, description of processes, activities, tasks, template, checklist, reference to standards, etc.

This Deployment Package (DP) supports the Basic Profile as defined in ISO/IEC TR 29110-5-6-2, the Management and engineering guide [ISO/IEC 29110]. The Basic Profile is one profile of the Generic profile group. The Generic profile group is applicable to VSEs that do not develop critical systems. The Generic profile group is composed of 4 profiles: Entry, Basic, Intermediate and Advanced. The Generic profile group does not imply any specific application domain. The Basic profile is targeted to VSEs working on one project at a time.

The Basic profile is composed of two processes: the Project Management Process and the System Definition and Realization Process.

The processes, activities and tasks described in this DP are consistent with those listed in ISO/IEC TR 29110 5-6-2 Systems Engineering — Lifecycle Profiles for Very Small Entities (VSEs) — Part 5-6-2: Management and engineering guide – Generic profile group: Basic profile.

The INCOSE Systems Engineering Handbook [INCOSE] has been used to develop this DP. The INCOSE Handbook is consistent with ISO/IEC 15288:2008 – *Systems and software engineering – System life cycle processes* [ISO 15288].

Information contained in this DP is applicable to VSEs that do not develop critical products that require intense verification and validation (V&V) activities. Those projects could use of the appropriate standards and guides (e.g. ANSI/GEIA EIA-632, MIL-STD-499, etc.)

This document is intended to be used by a VSE to establish processes to implement any development approach or methodology including, e.g., agile, evolutionary, incremental, test driven development, etc. based on the organization or project needs of a VSE.

The content of this document is entirely *informative*.

Once published by ISO, ISO/IEC TR 29110-5-6-2 will be available at no cost on the following ISO site: <http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html>

## Why is Systems Engineering important?

The way to effective Systems Engineering (SE) is not “in the direction of formal, formidable, massive documentation” [Chase]. Systems Engineering is a perspective, a process, and a profession [INCOSE]. SE has an iterative nature that supports learning and continuous improvement. SE has a horizontal orientation which means SE is a mechanism to establish agreements for the creation of products and services in a web of contractors and subcontractors. Therefore SE is the link between contractors, and PM, and the organizational parts of enterprises and single technical disciplines (e.g. Software, mechanics, HMI, EMC, etc.).

## Why is cooperation between Systems Engineering and Project Management important?

Systems engineers and program managers bring unique skills and experiences to the programs on which they work. There is also a “shared space” (PM/SE) where program managers and systems engineers collaborate to drive the program team’s performance and success. Therefore they have to collaborate.

Figure 1 shows a concept how systems engineering (SE) and project management (PM) might relate to each other. The basis for this concept is the project lifecycle as proposed by *ISO 21500 – Guidance on project management*. But it’s too simple just to consider the pure project time span for a product development. SE has to consider the whole life cycle of a product in the product concepts until product disposal. Therefore SE has to contribute in all project control activities and provide relevant inputs.



Figure 1 Overview of a concept for SE – PM cooperation

Because a deployment package is not a complete process reference model a VSE might need guidance about how they might perform a project.

To consider the idea for the ISO/IEC TR 29110 simplified technical processes have been defined (see the 9 coloured blocks in Figure 2). Each of these blocks consists of business aspects and technical aspects. Just the degree of involvement for PM and SE changes. Interface management or requirements engineering are commonly understand as SE activities. But they are also influenced by business aspects, enterprise interests or simply by available resources which are more in the PM domain. Therefore the addressed technical processes in figure 2 might be understood as common (PM&SE) activities.

Configuration management (CM) might be understood as an enterprise oriented task and used in every project. The activities of CM should start with the earliest project activities (the first idea for a project) and will not end with a project. The stored information must be available after a project is finished for several purposes (e.g. following project, legal issues, etc.).

Each of the technical process blocks includes activities which might be performed in different project phases. Figure 2 shows an example to map project process steps to single technical processes. The details for the technical processes are described in different DP packages.



Figure 2 DP structure and linkage to project steps

## Why is Functional Analysis and Physical Architecture Important?

The Functional Analysis and Physical Architecture design is performed during the development phase of the project (see Figure 1). Looking deeper into the development activities depicted in Figure 3 you can see that the refinement and allocation of requirements as well as identifying the design solution are tightly coupled. That is where the Functional Analysis and Architectural Design activities are taking place.

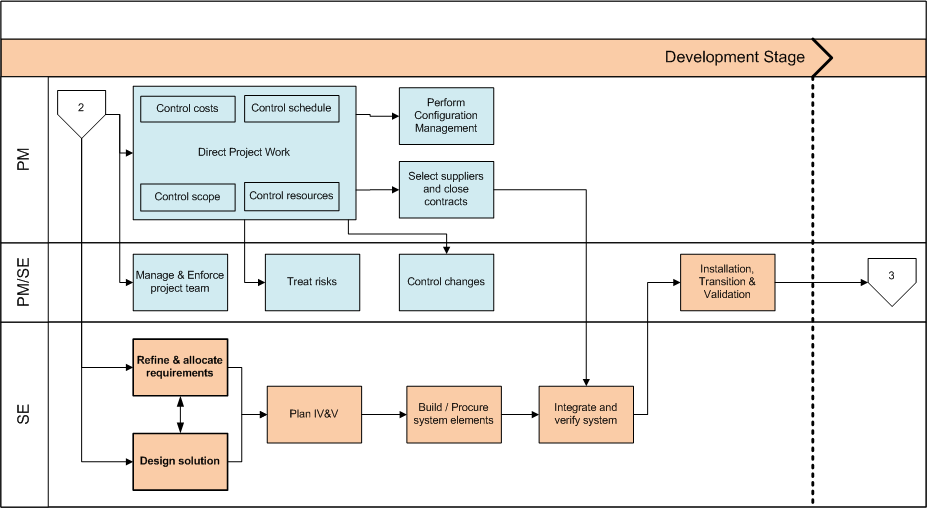


Figure 3 Overview of a concept for SE –Development Stage

During the Functional Analysis, which has to be seen as a part of the “Refine and allocate requirements” activity, the stakeholder needs and stakeholder requirements (not just the customer!) are analyzed to identify the system functions. These system functions subsequently are grouped into functional entities and their logical collaboration (among them and the environment) is described. It is important to notice that the Functional Analysis as such is an implementation free activity, allowing to completely understanding the problem space before selecting a possible implementation (solution). The Functional Analysis is therefore closely related to the Requirements Analysis process as described by ISO/IEC15288 allowing defining a set of system requirements.

The resulting logical architecture provides the foundation for the definition of the physical system architecture as defined during the Architectural Design process described by ISO/IEC15288.

During the design activities a physical architecture is identified and selected, implementing the functional entities identified before.

Design decisions, such as implementation of functions by separate components (e.g. computers) and the selection of specific interface types are taken during the definition of the Physical Architecture.

Note, that such design decisions might also require additional (derived) functionality, e.g. bus controller capability, which requires the return to the Functional Analysis activity for completion. Systems Engineering is an iterative business and in the least cases straight forward.

Finally it has to be mentioned that the aspects of the functional analysis and the physical architecture design already have to be considered and documented during the planning phase (refer to Figure 1). They are typically documented in the SEMP (which is part of the PM plan). A template on the structure of such a technical planning document can be found in 5.1.

## Tailoring this Deployment Package

This DP describes the minimum set of Functional Analysis and Physical Architecture design activities and tasks that should be implemented by a VSE. A VSE may have existing processes that can be substituted for these activities, tasks, steps, products and roles.

# Definitions

In this section, the reader will find two sets of definitions. The first set defines the terms used in all Deployment Packages, i.e. generic terms. The second set of terms used in this Deployment package, i.e. specific terms.

## Generic Terms

***Process***: a set of interrelated or interacting activities which transform inputs into outputs [ISO 9000:2000].

***Activity:*** a set of cohesive tasks of a process [ISO 15288]

***Task:*** required, recommended, or permissible action, intended to contribute to the achievement of one or more outcomes of a process [ISO 15288]

***Sub-Task:*** When a task is complex, it is divided into sub-tasks.

***Step:*** In a deployment package, a taskis decomposed in a sequence of steps.

***Role***: a defined function to be performed by a project team member, such as testing, filing, inspecting, coding [PMBOK].

***Product:*** result of a process [ISO 9000:2005]

NOTE There are four agreed generic product categories: hardware (e.g. engine mechanical part), software (e.g. computer program), services (e.g. transport), and processed materials (e.g. lubricant). Hardware and processed materials are generally tangible products, while software or services are generally intangible. Most products comprise elements belonging to different generic product categories. Whether the product is then called hardware, processed material, software or service depends on the dominant element.

***Artifact:*** information, which is not listed in ISO/IEC 29110 Part 5, but can help a VSE during the execution of a project.

## Specific Terms

RAMS: Reliability, Availability, Maintainability and Safety

SEMP: Systems Engineering Management Plan

The SEMP is the top-level plan for managing the systems engineering effort. The SEMP defines how the SE portion of the project will be organized, structured and conducted to provide a product that fulfills stakeholder needs. The SEMP is used for the technical management of the project. A SEMP outline is provided in INCOSE SE Handbook [INCOSE].

SysML: Systems Engineering Modelling Language (OMG®)

Architecture: A set of fundamental concepts and properties in a specified environment, embodied in elements, relationships and principles to guide system design and evolution NOTE Systems architecture provides the conceptual definition of the logical, physical structures, behavior, temporal relationships, and/or other aspects of a system and allocations among alternatives (e.g., physical elements, software, and operations; and/or functions in system-of-interest versus enabling system). [ISO/IEC/IEEE 42010]

Logical Architecture: (contraction of Logical view of the Architecture of the system) the logical view of the architecture of a system is composed of a set of related technical concepts and principles that support the logical operation of the system. It includes at the minimum a functional view, a behavioural view, and a temporal view. [ISO 15288]

Physical Architecture: (contraction of Physical view of the Architecture of the system) a physical view of the architecture is an arrangement of system elements and physical interfaces which provides the design solution for a product, service, or enterprise, and is intended to satisfy logical architecture elements and system requirements. It is implementable through technologies. [ISO 15288]

# Relationships with ISO/IEC 29110

This deployment package covers the processes and activities related to Functional and Physical Architecture for Very Small Entities (VSEs) as described in ISO/IEC 29110-5-6-2.

In this section, the reader will find a list of processes, activities, tasks and roles of ISO/IEC 29110-5-6-2 that are directly related to Functional Analysis and Physical Architecture design. Other activities might be directly impacted by the work result but are not explicitly mentioned here.

Tasks and steps elaborated in chapter 4 will link to the here identified ISO/IEC 29110-5-6-2 activities.

* **Process: System Definition and Realization (SR)**
* **Activity: SR.1 System Definition and Realization Initiation**
* **Tasks:**

|  |  |
| --- | --- |
| **SR.1.2 Define the data model of the project**  Define the entities to manage in the project (e.g. requirements, system element, IVV plan, IVV procedure, IVV results), their properties (e.g. maturity, version, target release) and their relation (e.g. satisfies, allocated to, verify, validate) | PM  WT |

* **Activity: SR.2 System Requirements Engineering (SR.O2, SR.O6, SR.O7)**
* **Tasks:**

|  |  |
| --- | --- |
| **Tasks** | **Roles** |
| **SR.2.1 Elicit acquirer and other stakeholders requirements and analyze system context**  Identify and consult information sources of requirements (Acquirer, users, stakeholders, previous systems, documents, etc.): Statement of Work, Concept documents, previous System description, etc.  Analyze the context of use of the system with acquirer and other stakeholders:   * Identify the stakeholders * Define the concepts of use of the system * Define scenarios, business processes   Generate or update the Concept of Operations that describes the way the system works from the operator’s perspective.  Identify and analyze requirements to   * Determinate the scope and system boundary, * If applicable, identify the strengths and weaknesses of the previous system * Ensure that the Stakeholder requirements are complete and consistent * Elicit missing Stakeholder requirements   Resolve conflicting, duplicate and out-of-scope Stakeholder requirements. Generate or update the Stakeholders’ *Requirements* *Specification*. | SYS  ACQ  STK |
| **SR.2.4 Elaborate System Requirements and the Interfaces**  Define the system boundary.  Define interface requirements between the System and its environment.  *Note: Interface requirements are included in System Requirements Specification. Separate specification document can be established.*  Define System requirements, System design constraints and interface requirements with external entities/actors using the SMART criteria: Specific, Measurable, Accepted, Realistic and Traced.  Define the external functions ensured by the system (black box).  Define reuse constraints.  Define the applicable requirements and constraints to the system  Generate or update the System Requirements Specification | SYS  DES |
| **SR.2.5 Elaborate System Elements Requirements Specifications and the interfaces**  *Note: System Element requirements are generally elaborated in parallel with the System Logical and Physical Architectural Design Activity (see Activities SR. 3.1 and SR. 3.3)*  Allocate System requirements to System elements in conformity with the functional and physical architecture and decompose requirements so that System element requirements are distinctively and clearly defined. Elaborate System element requirements derived from the System architectural design but that cannot be traced to a specific parent System requirement  Refine as necessary external interface requirements and identify internal interface requirements between System Elements.  Generate or update a System Element Requirements Specification for each System Element defined in the System Design Document.  *Note: interface requirements are included in System Elements Requirements Specifications. Separate specification document can be established.*  *Note: System elements requirements become needs and expectation in input of the system elements implementation.* | DES  SYS |
| **SR.2.8 Define or update traceability between Requirements**  According to the data model defined in SY1.2, at each level of decomposition of the system, define or update traceability between   * System requirements, interface requirements and their parent stakeholder’s requirements * System elements requirements, interface requirements and their parent system requirements. | SYS  DES |

* **Activity: SR.3 System Architectural Design (SR.O3, SR.O6, SR.O7)**
* **Tasks:**

|  |  |
| --- | --- |
| **Tasks** | **Roles** |
| **SR.3.1 Document or update the Logical System Design**  Elaborate the logical architecture with the internal functions of the system and their relations (interfaces), by analyzing:   * The System Requirements * The external functions of the system (black box)   Define the internal functions and interfaces.  Identify the artifacts to reuse. Decide whether to make, buy or reuse.  Define in parallel the System elements requirements and interface requirements | DES |
| **SR.3.2 Make trade-offs of the System Logical Architecture**  Make trade-offs among the different possible logical architectures relative to the requirements.  Update the Justification Document and establish traceability with the requirements as defined in SR1.2.  Logical architecture can be done in a model based environment and generated as a document. | SYS  DES |
| **SR.3.3 Document or update the Physical System Design**  Elaborate the physical architecture by:   * analyzing the System Requirements (e.g. non functional requirements allocated directly the System Elements) * analyzing the Logical Architecture and allocating internal functions to System Elements * Identifying System Elements to reuse.   Identify the artifacts to reuse. Decide whether to make, buy or reuse.  Analyze the design as needed to demonstrate it can satisfy System Requirements (e.g. maintainability, reliability, security, safety integrity, usability, etc.)  Elaborate the physical and functional interfaces (external and internal) between System Elements. Define in parallel the interface requirements. | DES |
| **SR.3.4 Make trade-offs of the System Physical Architecture**  Make trade-offs among the different possible physical architectures relative to the requirements and the logical architecture.  Update the Justification Document and establish traceability with the requirements.  Physical architecture can be done in a model based environment and generated as a document  Generate or update the Traceability Record. | SYS  DES |

# Description of Processes, Activities, Tasks, Steps and Products

## Plan the systems engineering approach

|  |  |
| --- | --- |
|  | |
| ***Refers to:*** | Task SR.1.2: Define the data model of the project |
| ***Objectives:*** | The systems engineering approach shall be captured in a SEMP (part of the Project Plan) and provide guidance and justification for the systems engineering activities. |
| ***Rationale:*** | Systems engineering is a planned activity, involving analytic and systematic thinking. |
| ***Roles:*** | Technical Leader / (Senior-) Systems Engineer (SYS) |
| Project Manager (PM) |
| Customer (ACQ) |
| ***Products:*** | Project Plan |
| PM-Plan (SEMP part) |
| ***Artifacts:*** |  |
|  |
| ***Steps:*** | 1.Define functional analysis approach |
| 2.Define the trade-off studies to be conducted and usage of prototypes |
| 3. Approve SEMP |
| ***Step Description:*** | ***Step 1.*** Define functional analysis approach esp. the usage of models during the development.  Define the approach used, for performing the functional analysis.  If models are used during for functional analysis, the following aspects should be described:   * scope and purpose, * how to setup traceability to requirements and other artifacts * how to perform configuration control * baseline and handover strategies (model or derived documentation)   Models in this context can be graphical models (e.g. SysML), mechanical models (e.g. CAD) or mathematical models (e.g. Matlab Simulink)  ***Step 2.*** Define the included trade-off studies to be conducted and the usage of prototypes   * Define in what key aspects trade-off studies will be conducted for. * For envisaged prototypes clearly define the scope and purpose of such prototypes, when they will be build and how the derived design decisions will be documented.   ***Step 3.*** Approve SEMP   * Obtain customer sign-off on systems engineering approach.   Note: The described activity only focuses on the SEMP parts relevant for the Functional Analysis and Physical Architecture design activities! |

## Identify system functional input requirements and needs

|  |  |
| --- | --- |
|  | |
| ***Refers to:*** | SR.2.1 Elicit acquirer and other stakeholders requirements and analyze system context |
| ***Objectives:*** | Identify system functional and performance input requirements and needs from all stakeholders that may have an impact on the functionality the system has to provide. |
| ***Rationale:*** | Minimization of the risk of iterations late during the development cycle. Missings during this stage usually show up late during system validation! |
| ***Roles:*** | Project Manager (PM) |
| Technical Leader / Systems Engineer (SYS) |
| Customer (ACQ) |
| ***Products:*** | Stakeholder Requirements Specification |
| Concept documentation (Draft) |
| ***Artifacts:*** |  |
|  |
| ***Steps:*** | 1.Capture the customer needs |
| 2. Analyze Concept of Operation |
| 3. Analyze company policies and specialty engineering needs |
| 4. Analyze Standards and Regulations |
| ***Step Description:*** | ***Step 1.* Capture the customer needs**  Identify all operational needs, e.g. expressed in the statement of work, customer interviews or operational concept papers  Capture all stakeholder (functional or performance) requirements expressed by the customer.  ***Step 2.* Analyze Concept Documentation (if provided by customer)**  The operational concept documents (refer [ANSI G-043A-2012] describe the system characteristics for a proposed system from the users viewpoint. Therefore it is a great source of information to understand what the system is needed to perform. Besides the well known operational concept also other concept documentation, e.g. concept of production, deployment, support and disposal, need to be considered.  Note: If not provided by customer, consider to setup a draft operational concept document, capturing the here identified high level operational needs from customer perspective. It serves to the validation of the elicited set of system requirements at a later stage.  ***Step 3. Analyze company policies and* specialty engineering needs**  Check company internal policies as well as the specialty engineering domain, e.g. RAMS, for needs, that directly require certain behavior of the intended system.  ***Step 4. Analyze* Standards and Regulations**  Check domain specific standards and regulations for required functionality the intended system must provide. |

## Identify System Boundary and Use Cases

|  |  |
| --- | --- |
|  | |
| ***Refers to:*** | SR.2.1 Elicit acquirer and other stakeholders requirements and analyze system context  SR.2.4 Elaborate System Requirements and the Interfaces |
| ***Objectives:*** | Identify the system (operational) boundary and identify the system use cases. |
| ***Rationale:*** | It is crucial to identify what is in scope and out of scope for the system of interest. |
| ***Roles:*** | Technical Leader / Systems Engineer (SYS) |
| Design Team (DES) |
| ***Products:*** | System Design Document |
|  |
| ***Artifacts:*** |  |
|  |
| ***Steps:*** | 1.Identify system boundary and interfacing entities |
| 2.Identify system use cases |
| 3. Validate use cases against requirements |
| ***Step Description:*** | ***Step 1.*** Identify system boundary and interfacing entities   * Define the system boundary and identify the functionally interfacing entities (in the following called: actor), e.g. human actors, systems, equipments.   ***Step 2.*** Identify system use cases   * Identify, based on the Stakeholder Requirements Document and the Concept of Operation the system use cases. A system use case describes a high level usage (“operational thread”) of the system. Describe the scope of each use case. For later reference link the actors to the functionally related use cases.   Note: A use case does not imply the system’s internal structure.  Note: The use cases are the starting point for the following functional analysis. Therefore they should be complete enough to allow the identification of required functionality but small enough to manage its complexity. “Overlapping” use cases are not a problem, since this would just lead to multiple the identification of a function.  ***Step 3.*** Validate use cases against requirements   * Check that all functional input requirements and needs are traced to system use cases. If necessary extend the scope of use cases.   Note: The Functional Analysis in principal is an activity that can be very well conducted using a modeling language, like SysML. That is why some SysML specific terms are used to describe the activities. Nevertheless, this task can be performed in a textual manner. In the latter case special care should be taken on how to ensure completeness, consistency of the design documentation. |

## Describe use case with operational scenarios and related interfaces

|  |  |
| --- | --- |
|  | |
| ***Refers to:*** | SR.2.1 Elicit acquirer and other stakeholders requirements and analyze system context |
| ***Objectives:*** | Capture the system functions and logical interfaces thru the elicitation of operational scenarios. |
| ***Rationale:*** | Assessment of the identified use cases and capturing the related operational scenarios associated to it. |
| ***Roles:*** | Systems Engineer (SYS) |
| Design Team (DES) |
| Project Manager (PM) |
| ***Products:*** | System Design Document |
| System Requirements Specification |
| Statement of Work (Updated) |
| Concept of Operations (Updated) |
| ***Artifacts:*** |  |
|  |  |
| ***Steps:*** | 1. Capture operational scenarios |
| 1. Derive logical interfaces |
|  | 1. Define system states and modes |
|  | 1. Link and validate stakeholder requirements |
|  | 1. Update and approve Stakeholder Requirements |
|  | 1. Elicit System Requirements |
| ***Step Description:*** | ***Step 1.* Capture operational scenarios**  Describe the operational scenarios per system use case. Identify the logical flows of system functions and required interactions with the actors. Typically this step is done using activity and sequence diagrams.  Note: A system function is to be seen on an abstract level, i.e. it has to match with its granularity the functional system requirement that will be derived from it on a later stage.  Note: This part of the functional analysis is performed with the system looked at in a “black box” manner.  ***Step 2.* Derive logical interfaces**  From the operational scenarios compile the system logical interactions between the system and the actors into a separate view. E.g. the SysML related diagram is the Internal Block diagram (IBD) showing the entities, their ports and connections. Check the identified interfaces against the allocations of actors with use cases.  ***Step 3.* Define system states and modes**  The system states and modes describe when the system will provide certain functionality and how the system will react based on the system history. Use this approach to identify missing functions, e.g. to avoid dead lock situations.  ***Step 4.* Link and validate stakeholder requirements**  Link the identified system functions and interfaces to the stakeholder requirements they correspond to. Check that   * All system functions are covered by stakeholder requirements (otherwise important and potentially costly aspects are not contractually recognized) * All functional stakeholder requirements are linked to a system function. Identify requirements out-of-scope * All stakeholder requirements are contradiction free. * At this point the statement of work might need to be re-negotiated due to the findings made.   ***Step 5.* Update and approve Stakeholder Requirements**  The identified insufficiencies should lead to an update of the Stakeholder Requirements Document and Statement of Work, to be approved by the customer.  Note: The identified operational scenarios are a good opportunity for an early validation of the customer operational needs. Update/extend the operational concepts with the identified operational scenarios.  ***Step 6.* Elicit System Element Requirements**  From the defined system black-box behavior elicit a coherent set of functional, performance and interface requirements. Document these in the System Requirements document. Establish traceability and check completeness. |

## Identify Logical Architecture

|  |  |
| --- | --- |
|  | |
| ***Refer to:*** | SR.3.2 Make trade-offs of the System Logical Architecture |
| ***Objectives:*** | Identify logical entities of the system for function allocation. |
| ***Rationale:*** | A suitable logical architecture has to be identified for later function allocation. |
| ***Roles:*** | Systems Engineer (SYS) |
| Design Team (DES) |
| ***Products:*** | System Design Document |
| Justification Dossier |
| ***Artifacts:*** |  |
|  |
| ***Steps:*** | 1. Identification of candidate solutions |
| 1. Document design justification |
| 1. Capture System Logical Architecture |
| ***Step Description:*** | ***Step 1. Identification of candidate solutions***  For the identification of the most suitable logical architecture a set of considerations have to be made, e.g.:   * Intended /given physical architecture (logical items have to be allocated to physical system elements later) * Legacy systems/components * Re-use and modularity * Interfaces, their performance and capacity * Safety aspects   Define the assessment criteria and weight the importance of each aspect.  Based on these considerations a set of possible logical architectures can be identified. Perform a trade-off study evaluating the assessment criteria for each candidate solution.  ***Step 2.* Document design justification**  Include the design justification for the selected logical architecture into the Justification Dossier.  ***Step 3.* Capture System Logical Architecture**  Capture the Logical Architecture in a diagram and incorporate it into the System Design Document. The appropriate SysML diagram would be the Block Definition Diagram.  Note: The level of decomposition depends on the next documentation / integration level of the system. Envisage to have a separate System Design Document / system model for each decomposition level. |

## Allocate functions to logical architecture

|  |  |  |
| --- | --- | --- |
|  | | |
| ***Refer to:*** | SR.2.5 Elaborate System Elements Requirements Specifications and the interfaces  SR.2.8 Define or update traceability between Requirements  SR.3.1 Document or update the Logical System Design |
| ***Objectives:*** | Allocate functions and external interfaces to logical entities.  Identify internal interfaces.  Elaborate System Element Requirements |
| ***Rationale:*** | Logical entities are linked to physical system elements for implementation. |
| ***Roles:*** | Systems Engineer (SYS) |
| Design Team (DES) |
| ***Products:*** |  |
|  |
|  |
| ***Artifacts:*** | *System Design* documentation |
| *System Elements Requirements Specification* |
| ***Steps:*** | 1. Elaborate white box operational scenarios |
| 1. Derive logical (internal) interfaces |
| 1. Define system element states and modes |
| 1. Document Logical Architecture |
| 1. Elicit functional System Element Requirements |
| ***Step Description:*** | ***Step 1.* Elaborate white box operational scenarios**  Extend the previous black-box operational scenarios in such way, that the functions are allocated to the respective logical element.  For activity diagrams you want to use an “Allocation Activity Partition” (also known as swimlanes). Sequence diagrams are getting extended in the way that additional lifelines per logical element are getting added to the diagram and the functions and message exchanges are moved onto those elements.  Note that functions might need to be decomposed to be uniquely allocated (which results in a system element requirements decomposition later as well).  ***Step 2.* Derive logical (internal) interfaces**  While the external interfaces as such are unchanged, besides the fact that they are now allocated to a logical element, the internal interfaces need to be identified.  First indications of a message exchange between logical elements are the flows reaching from one “Allocation Activity Partition” into another. On the sequence diagrams a former sequential row of functions that is now allocated to different logical elements requires messages exchanges between the lifelines in order to reestablish the sequence of functions. Complete those messages.  From the white box operational scenarios compile the system external and internal logical interactions of the logical elements and the actors into a separate view (Internal Block diagram (IBD)).  ***Step 3.* Define system element states and modes**  The logical system element states and modes describe when individual system element will provide which functionality and how the system elements collaborate in order to produce the expected system behavior. Use this approach to identify missing functions, e.g. to avoid dead lock situations. Check the defined behavior against the black-box model.  Note: Some SysML tools today provide the capability of model execution, allowing to check the consistency of your model.  ***Step 4.* Document Logical Architecture**  Document the logical architecture, allocation of functions and interfaces in the System Design Document (update of results from 4.5).  ***Step 5.* Elicit System Element Requirements**  From the defined system white-box behavior elicit a coherent set of functional, performance and interface requirements. They might be either a flow down of system requirements but might have to be decomposed (due to function decomposition) or refined.  Document these in the System Element Requirements Specification. Establish traceability and check completeness.  Note: A functional system element requirement might not be a child of a higher level system requirement; it is a so called “derived” requirement. Nevertheless functional requirements should always be traceable to a system design element, e.g. function, property or interface, i.e. it is always derived from the design and is therefore justified.  Note: If the SE approach is Model-Based (MBSE), the SEMP should identify, if a direct traceability from system requirement to system element requirement is required. It might be more efficient (and as effective) to establish a traceability through the design model.  Care should be taken not to hide design and implementation issues for a particular system element within a requirement. Such constraints should be clearly segregated from the requirements itself. |

## Identify Physical Architecture

|  |  |
| --- | --- |
|  | |
| ***Refers to:*** | SR.2.5 Elaborate System Elements Requirements Specifications and the interfaces  SR.3.3 Document or update the Physical System Design  SR.3.4 Make trade-offs of the System Physical Architecture |
| ***Objectives:*** | Develop the adequate Physical Architecture based on the Logical Architecture including further requirements and constraints. |
| ***Rationale:*** | The physical system elements are the ones going to be implemented. |
| ***Roles:*** | Systems Engineer (SYS) |
| Design Team (DES) |
| ***Products:*** | System Element Specification |
| System Design Document |
| Justification Dossier |
| ***Artifacts:*** |  |
|  |
| ***Steps:*** | 1. Synthesize Physical Architecture |
| 1. Allocate Functional Architecture |
| 1. Document Design Justification |
| 1. Validate Physical Architecture |
| 1. Elicit non-functional System Element Requirements |
| ***Step Description:*** | ***Step 1.*** **Synthesize Physical Architecture**  Based on the Logical Architecture and other constraints, like already existing or COTS elements, different physical architectural solutions might be identified and assessed. This task includes the identification of physical elements able to implement the logical elements defined in the Logical Architecture as well as physical interfaces being able to implement the logical interfaces.  To be able to find the right implementation solution, analysis on certain aspects of the system might need to be performed using prototypes or simulations to gather enough information.  A trade-off study may help to identify the solution, which will be implemented.  Depending on the level of the Physical Architecture it needs to be decided whether a system element shall be a make or buy item. If it is a make item, further decide if it is implemented in hardware or in software or if a further decomposition level needed.  While the functional analysis was implementation free, the selection of a physical architecture is a design decision might reveal the need for additional (derived) functionality. Consequently the functional architecture need to be refined (introduction of additional use cases) to reflect this additional need (example: bus controller function). (refer to 4.3)  ***Step 2.* Allocate Functional Architecture**  Allocate the logical elements to the elements of the physical architecture.  Note: A complete mapping assures that the physical architecture is compatible with the logical architecture. Each logical elements needs to be mapped on a single configuration item within the physical architecture. If this is not possible, the logical element needs to be further refined and decomposed (refer 4.5). A single element of the physical architecture may implement more than one logical element.  Interfaces defined by the logical architecture crossing physical element borders are allocated to the physical interfaces. From the specified items, events and data exchanged on an interface the data transfer requirements (amount of data, transfer rate) can be evaluated. The selected type of interface needs to comply with these data transfer requirements. Logical interfaces remaining inside a physical element might not require a physical interface (design decision for the implementing level).  ***Step 3*. Document Design Justification**  Justification for the selected physical architecture needs to be documented in the Design Justification dossier, in order to provide a rationale for the derived requirements. Include also a summary of the performed analysis.    ***Step 4.* Validate Physical Architecture**  Check that the elements of the selected physical architecture are able to implement the allocated logical elements e.g. in terms of performance. This might already be checked during the first step.  For validation of the selected Physical Architecture and to verify that non-functional requirements are considered, look into the following analysis results:   * Safety Analysis * Performance Analysis * Reliability Analysis * Maintainability Analysis * Testability Analysis * Environmental Analysis * Usability Analysis * Growth Capacity * others   The results of the performed analysis and tests are documented in the Justification Dossier for later requirements validation.  Detail the physical architecture in the System Design Document, i.e. the product breakdown structure, interface diagrams, system elements description, physical interface descriptions and the mapping of the logical architecture.  ***Step 5.* Elicit non-functional System Element Requirements**  All non-functional requirements are decomposed and allocated to the respective elements of the physical architecture.  Capture the physical interface requirements for each element. Compile all non-functional requirements in the System Element Specification. |

## Role Description

This is an alphabetical list of the roles, abbreviations and required competencies description tailored to the roles involved in this DP.

|  | **Role** | **Abbreviation** | **Competency** |
| --- | --- | --- | --- |
| 1. | Acquirer | ACQ | The Acquirer is the Stakeholders representative. He is responsible for the acquisition of the System.  The acquirer may be internal or external to the supplier organization. Acquisition of a software product may involve, but does not necessarily require, a legal contract or a financial transaction between the acquirer and supplier. In some context the Acquirer is the end user of the system.  Knowledge of the Stakeholders processes and ability to explain the Stakeholders requirements. The Acquirer is the role of the organization that receives the product or service. In some context the Acquirer is the end user of the system.  The Acquirer must have the authority to approve the requirements and their changes.  The Stakeholders includes user representatives in order to ensure that the operational environment is addressed.  Knowledge and experience in the application domain. |
| 2. | Designer | DES | Knowledge and experience in the architecture design.  Knowledge of the revision techniques.  Knowledge and experience in the planning and performance of integration tests.  Knowledge of the editing techniques.  Experience on the system development and maintenance. |
| 5. | Project Manager | PM | Leadership capability with experience making decisions, planning, personnel management, delegation and supervision, finances and system development. |
| 6 | Stakeholder | STK | Stakeholders are actors that have an interest in the system, all along its life cycle, such as, representatives of users, users, maintainers, security, trainers, regulatory bodies, suppliers.  STK should have Knowledge of the Stakeholder (e.g. manufacturer, maintainer, tester, logistic) processes and ability to explain the Stakeholder requirements.  The Stakeholder (representative) must have the authority to approve the requirements and their changes.  Knowledge and experience in the application domain. |
| 8. | Systems Engineer | SYS | Knowledge and experience eliciting, specifying and analyzing the requirements.  Knowledge in designing user interfaces and ergonomic criteria.  Knowledge of the revision techniques.  Knowledge of the requirements authoring.  Knowledge of the business domain  Experience on system development, integration, operation and maintenance  Experience on the system development and maintenance. |
| 9. | Work Team | WT | Knowledge and experience according to their roles on the project: SYS, DES, DEV, IVV.  Knowledge on the standards used by the Acquirer and/or by the VSE. |

## Product & Artifacts Descriptions

This is a list of the input, output and internal process products & artifacts, its descriptions, possible states and the source of the product.

|  | **Name** | **Description** | **Source** |
| --- | --- | --- | --- |
| *1.* | *Statement of Work* | It may Include:   * *Product Description* * *Scope* * *Objectives* * *Deliverables*   The applicable status is:reviewed | Customer |
| *2.* | *System Design Document* | This document includes textual and graphical information on the system structure and behavior. This structure may include the following parts:  Architectural High Level System Design – Describes the overall *System* structure (logically and physically):   * Identifies the required system elements * Identifies the relationship between system elements * Identifies the collaboration of system elements * Consideration is given to any required: * system performance characteristics * system interfaces * security characteristics * database design requirements * error handling and recovery attributes   Detailed Low Level System Design – includes details of the system elements to facilitate its construction and test within the construction environment;   * Provides detailed design (could be represented as calculations, drawings, specifications, and data sheets) * Provides characteristics of inputs / outputs * Provides specification of storage needs * Establishes required naming conventions * Defines the characteristics of structures * Defines components and their purpose * Provides the engineering / procurement specifications   The applicable statuses are:verified and baselined. | Functional Analysis  and  Physical Architecture |
| *3.* | *Requirements Specification*  *(StRS)*  *(SyRS)*  *(SyElRS)* | Includes an introduction and a description of the requirements. It may contain:  Requirements description:   * functionality – established needs to be satisfied by the system when it is used in specific conditions. Functionality must be adequate, accurate and safe. * user interface – definition of those user interface characteristics that allow to understand and learn the system easily so the user be able to perform his/her tasks efficiently including the interface exemplar description; external interfaces – definition of interfaces with other systems or software; * reliability – specification of the system execution level concerning the maturity, fault tolerance and recovery; * efficiency – specification of the system execution level concerning the time and use of the resources; * maintenance – description of the elements facilitating the understanding and execution of the future system modifications; * safety – specification of constraints avoiding unintended damage to human beings or the environment * security - * portability – description of the system characteristics that allow its transfer from one place to other; * design and construction limitations – needs imposed by the customer; * inter-operability – capability for two or more systems or system components be able to interact with each other and use it. * reusability – feature of any product/sub-product, or a part of it, so that it can be used by several users as an end product, in the own system development, or in the execution of other system products. * legal and regulative – needs imposed by laws, regulations, etc.   Each requirement is identified, unique and it is verifiable or can be assessed.  The applicable statuses are:verified, validated and baselined. | Functional Analysis  and  Physical Architecture |
| *4.* | *Justification Dossier* | The justification dossier contains a collection of analysis results, assumptions, constraints and other reasoning’s for the decisions taken during the Systems Engineering activities. |  |
| *5.* | *Concept Documentation* | **Concept of Production** – Describes the way the system will be manufactured, including any hazardous materials used in the process.  **Concept of Deployment** – Describes the way the system will be delivered and installed.  **Concept of Operations (ConOps)** – Describes the way the system works from the operator’s perspective. The ConOps includes the user description and summarizes the needs, goals, and characteristics of the system’s user community. This includes operation, maintenance, and support personnel.  **Concept of Support** – Describes the desired support  infrastructure and manpower considerations for maintaining the system after it is deployed. This includes specifying equipment, procedures, facilities, and operator training requirements.  Source: [INCOSE]  For more information see [ANSI G-043A-2012]. |  |
| *6.* | *PM-plan*  *(SEMP aspects)* | The Systems Engineering Management Plan (SEMP) provides the framework and guidance for all engineering activities within the project.  The SEMP describes the activities, products, processes, tools and controls that are used during the development phases of the project.  The SEMP is a living document and should be updated in regular intervals to reflect the necessary changes as the project evolves. |  |

# Templates

## Template: Systems Engineering Management Plan (SEMP)

A good template for a SEMP can be found in the [G2SEBOK], contributed by Eric Honour. Following the headline structure of such template:

1. INTRODUCTION
   1. Overview
   2. System Description
2. REFERENCED DOCUMENTS
3. TECHNICAL PROGRAM PLANNING AND CONTROL
   1. Task Descriptions
      1. Definition of Work
      2. Subcontractor Work Effort
      3. Schedule
   2. Organization
      1. Technical Organization
      2. Technical Functions
      3. Program Interfaces
   3. Technical Control
      1. Requirements Management
      2. Technical Issues Management
      3. Risk Management
      4. Interface Control
      5. Configuration Control
      6. Document Control
   4. Performance Control
      1. Technical Performance Measures (TPM) Process
      2. Technical Performance Measures
   5. Program and Design Reviews
      1. Review Process
      2. Review Schedule
4. SYSTEM ENGINEERING PROCESS
   1. Process Description
      1. Operational Definition
      2. Requirements Definition
      3. System Architecting
      4. Preliminary Component Design
      5. Detailed Component Design
      6. Prototype Manufacture
      7. System Integration
      8. System Verification
      9. System Validation
      10. Operation and Maintenance
   2. Related Processes
      1. Electronics
      2. Software
      3. Mechanical
      4. Process Development
   3. Trade Studies
      1. Trade Study Process
      2. Trade Study Tasks
   4. Requirements Allocation
   5. Design Optimization/Effectiveness
      1. Analysis Methods and Tools
      2. Design Analyses
   6. Documentation
      1. Specification Tree
      2. Other Documents
      3. Document Generation Methods
5. ENGINEERING SPECIALTY INTEGRATION
   1. Control of Engineering Specialties
   2. Integrated Logistics Support Plan
   3. Reliability/Maintainability/Availability Plan
   4. Safety Plan
   5. Human Factors Engineering Plan
   6. Security Plan
   7. Electromagnetic Effects Plan
   8. Value Engineering Plan

5.X (Other Plans)

For the annotated description of the document outline refer to:

<http://g2sebok.incose.org/documents/assets/MSS/Final/semp_honour.pdf>

Another example outline of a SEMP can be found in the [ISI/IEC CD 24748-4.1]

## Template: System Requirements Specification

Following a generic template for a system requirements specification:

1. Introduction
   1. Purpose
   2. Scope
   3. System Overview
2. Functional Requirements
   1. Functions
   2. Performance
   3. Interfaces (logical / data)
3. Non-Functional Requirements
   1. Reliability
   2. Usability
   3. Availability
   4. Maintainability
   5. Security
   6. Safety
   7. Efficiency
   8. Portability
   9. Reusability
4. Environmental Conditions
   1. Shock
   2. Vibration
   3. Noise
   4. Electromagnetic Compatibility
   5. Electromagnetic Radiation
   6. Corrosiveness (salt spray)
5. Legal and regulative constraints
6. Constraints

## Template: System Design Document

Following a generic template for a system design document:

1. Introduction
   1. Purpose
   2. Scope
2. System Architecture
   1. Architectural Design Approach
   2. Architecture Design
3. Data Dictionary
4. System Domain Design
   1. System Domain Chart
   2. System Domains
      1. Domain X
         1. Component Y
            1. Component Task Z
5. Data Design
   1. Persistent/Static Data
      1. Persistent/Static Data Store X
   2. Transient/Dynamic Data
   3. External Interface Data
   4. Transformation of Data
6. User Interface Design
   1. User Interface Design Overview
   2. User Interface Navigation Hierarchy
   3. User Function Categories (or Use Cases)
      1. Function (or Use Case) X
         1. Screen/Report Format/Other User Interface XX
            1. Function (or Use Case) X Screen/Other User Interface XX Fields
7. Other Interfaces
   1. Interface X
8. Other Design Features
9. Requirements Traceability Matrix
10. References
11. Glossary
12. Revision History
13. Appendices

Source: [TPDF]

# Example

# Checklist

Review, Inspections, Testing Checklists

# Tools

Since the Functional Analysis and Physical Architecture design can be performed in a model based manner the question of suitable tools arises. Since tool capabilities are evolving constantly some considerations to be taken before selecting a tool:

**Modeling Method** – All UML/SysML tools basically allow setting up a systems model. In order not to generate another inconsistent set of diagrams, a modeling method needs to be defined and followed; such mandatory method gives guidance to the engineer as well as control during the modeling process. Therefore check how the tool environment is supporting your workflow before selecting a tool. Some methods are defined as a concept on paper, while others are deeper implemented by tools and supported by respective profiles and tool add-ins. Some of the today existing MBSE methods are:

* INCOSE Object-Oriented Systems Engineering Method (OOSEM),
* IBM Rational Telelogic Harmony-SE,
* Vitech MBSE Methodology,
* JPL State Analysis (SA),
* Dori Object-Process Methodology (OPM)
* Weilkiens Systems Modeling Process (SYSMOD)
* Fernandez Process Pipelines in OO Architectures (PPOOA)
* Alstom ASAP methodology

For more information on the methodologies refer to [MBSE Survey].

**Model consistency and execution** – ensure that the modeling tool of your choice is able to check for correctness and consistency of your model. Metrics for completeness (e.g. descriptions, requirements traceability) or complexity are beneficial.

**Traceability** – Check the modeling tool connectivity to the requirements management tool of your choice allowing to establish traceability and to perform bi-directional impact analysis.

**Configuration Control** – Check how far the modeling tool is supporting your needs for configuration and change management. Look for the appropriate connectivity to CM and CLM tools.

**Document generation** – a powerful document generation based on model data is a mandatory function, in order to take maximum benefit from the MBSE approach. Only then you will be able to concentrate on the system model baseline and generate your design documentation from it in no time. Using copy-&-paste for putting diagrams into documents might lead to a CM-nightmare.

**Usability / Tool Customization** – The SysML tools available today are mostly UML tools, adapted via a SysML profile. While this is sufficient for the adaptation of the modeling language itself, it does not adjust a powerful “software developer” tool to the systems engineers needs. The capability to limit the features and tools offered by the modeling tool directly translates into training and consulting effort necessary to master it.

A short and not comprehensive list of available SysML tools is:

* NoMagic MagicDraw
* Papyrus
* Sparxx Enterprise Architect
* IBM Rational Rhapsody Designer for Systems Engineers
* Atego Artisan Studio
* Modelio SysML Designer
* UModel
* TOPCASED-SysML

For more information refer to the INCOSE Tools Database Working Group (TDWG).

# References to Other Standards and Models

This section provides references of this deployment package to ISO/IEC 15288, ISO 9001, ISO 21500 and to the Capability Maturity Model IntegrationSM for Development version 1.3 of the System Engineering Institute (CMMI-DEV®[[1]](#footnote-1)).

Notes:

* This section is provided for information purpose only.
* Only tasks covered by this Deployment Package are listed in each table.
* The tables use the following convention:
* Full Coverage = F
* Partial Coverage = P
* No Coverage = N

## ISO/IEC 15288 Coverage Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| **Title of the Task and Step** | **Coverage**  **F/P** | **Clause of ISO/IEC 15288** | **Comments** |
| 4.1 | P | 6.3.1 Project Planning Process | SEMP not explicitly mentioned in 15288 |
| 4.2, 4.3, 4.4, 4.5, 4.6 | P | 6.4.2 Requirements Analysis Process | Also check DP Requirement Engineering |
| 4.7 | P | 6.4.3 Architectural Design Process | Also check DP Requirement Engineering |

## CMMI for Development, Version 1.3 Coverage Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| **Title of the Task and Step** | **Coverage**  **F/P/N** | **Objective/ Practice of CMMI** | **Comments** |
| 4.2, 4.3, 4.4, 4.5, 4.6, 4.7 | P | Requirements Development | Also check DP Requirement Engineering |
| 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7 | P | Technical Solution | Also check DP Requirement Engineering |

## ISO 21500 Coverage Matrix

|  |  |  |  |
| --- | --- | --- | --- |
| **Title of the Task and Step** | **Coverage**  **F/P/N** | **Clause of ISO 21500** | **Comments** |
| 4.1 | P | 4.3.3 Develop project plans | SEMP not explicitly mentioned in 21500 |

# References

|  |  |
| --- | --- |
| **Key** | **Reference** |
| [ISO/IEC 15288] | ISO/IEC 15288:2008 Systems and software engineering - System life cycle processes. |
| [ISO/IEC 24765] | ISO/IEC 24765:2011, Systems and Software Engineering Vocabulary.  An electronic version of the glossary is available at: <http://pascal.computer.org/sev_display/index.action> |
| [ISI/IEC CD 24748-4] | Systems engineering -- Application and management of the systems engineering process |
| [ISO/IEC 29110] | ISO/IEC TR 29110 5-6-2 Systems Engineering — Lifecycle Profiles for Very Small Entities (VSEs) — Part 5-6-2: Management and engineering guide – Generic profile group: Basic profile.  Once published by ISO, this document will be available at no cost on the following ISO site: <http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html> |
| [INCOSE] | Systems Engineering Handbook - A Guide for System Life Cycle  Processes and Activities, INCOSE (International Council on  Systems Engineering), Version 3.2.2, 2012  <https://www.incose.org/ProductsPubs/products/sehandbook.aspx> |
| [IEEE 1233] | IEEE Guide for Developing System Requirements Specifications |
| [ANSI G-043A-2012] | ANSI/AIAA Guide to the Preparation of Operational Concept Documents |
| [G2SEBOK] | Guide to the Systems Engineering Body of Knowledge -- G2SEBoK  <http://g2sebok.incose.org/> |
| [MBSE Survey] | INCOSE Survey of Model-Based Systems Engineering (MBSE) Methodologies (INCOSE-TD-2007-003-01), Jeff A. Estefan et.al. ,2008 |
| [TPDF] | TEXAS PROJECT DELIVERY FRAMEWORK, SYSTEM DESIGN DESCRIPTION ([www.dir.texas.gov](http://www.dir.texas.gov)) |

# Evaluation Form

|  |
| --- |
| **Deployment Package – Functional and Physical Architecture Version 0.1**  Your feedback will allow us to improve this deployment package, your comments and suggestions are welcomed. |
| **1. How satisfied are you with the CONTENT of this deployment package?**   *Very Satisfied*  *Satisfied*  *Neither Satisfied nor Dissatisfied*  *Dissatisfied*  *Very Dissatisfied* |
| **2. The sequence in which the topics are discussed, are logical and easy to follow?**   *Very Satisfied*  *Satisfied*  *Neither Satisfied nor Dissatisfied*  *Dissatisfied*  *Very Dissatisfied* |
| **3. How satisfied were you with the APPEARANCE/FORMAT of this deployment package?**   *Very Satisfied*  *Satisfied*  *Neither Satisfied nor Dissatisfied*  *Dissatisfied*  *Very Dissatisfied* |
| **4. Have any unnecessary topics been included? (please describe)** |
| **5. What missing topic would you like to see in this package? (please describe)**   * Proposed topic: * Rationale for new topic |
| **6. Any error in this deployment package?**   * + Please indicate:     - * Description of error :       * Location of error (section #, figure #, table #) : |
| **7. Other feedback or comments:** |
| **8. Would you recommend this Deployment package to a colleague from another VSE?**   *Definitely*  *Probably*  *Not Sure*  *Probably Not*  *Definitely Not* |

**Optional**

* Name:
* e-mail address : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Email this form to**: [joseph.marvin@incose.org](mailto:joseph.marvin@incose.org) or [claude.y.laporte@etsmtl.ca](mailto:claude.y.laporte@etsmtl.ca)

1. SM CMM Integration is a service mark of Carnegie Mellon University.

   ® Capability Maturity Model, CMMI are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University. [↑](#footnote-ref-1)