Deployment Package
Software Implementation
Entry Profile

Authors	G. Hernandez École de Technologie Supérieure (ETS), (Canada) W. Gonzalez - École de Technologie Supérieure (ETS), (Canada)
Editor	C. Y. LAPORTE – École de Technologie Supérieure (ETS), (Canada)
Creation date	25/06/2010
Last update	19/11/2010
Status	Draft
Version	0.2

Version History

Date	Version	Author	Modification
25/06/2010	0.1	G. Hernandez W. Gonzalez	Document Creation
19/11/2010	0.2	G. Hernandez W. Gonzalez	

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
SI	Software Implementation

Table of Contents

1. Technical Description

Purpose of this document

This Deployment Package (DP) supports the Entry Profile as defined in ISO/IEC TR 29110 Part 5-1-1, the Management and Engineering Guide [ISO/IEC29110]. The Entry Profile is one profile of the Generic profile group. The Generic profile group is applicable to VSEs that do not develop critical software. 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 Entry profile is targeted to VSEs working on small projects (e.g. at most six person-months effort) and for start-up VSEs. The Entry Profile provides a foundation for a migration to the Basic Profile Processes.

A 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: description of processes, activities, tasks, roles and products, template, checklist, example and tools.

The content of this document is entirely *informative*.

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.

Once published by ISO, ISO/IEC TR 29110-5-1-1 will be available at no cost on the following ISO site: http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html

Why Software Implementation is Important?

Implementation is the carrying out, execution, or practice of a plan, a method, or any design for doing something. As such, implementation is the action that must follow any preliminary thinking in order for something to actually happen.

In an information technology context, implementation encompasses all the processes involved in getting new software or hardware operating properly in its environment, including installation, configuration, running, testing, and making necessary changes. The word deployment is sometimes used to mean the same thing [crm.com].

The importance of having this process set lies in the systematic analysis and way to carry out the tasks. Every activity aim to reach the objectives stipulated in the statement of work and the main goal is to achieve the final product with all the attributes demanded by the customer with a high level of quality.

The importance of every software implementation activity is described as follows:

- Software implementation initiation allows to prepare the team work for the activities and to have all the necessary tools to accomplish the work.

- Software Requirements Analysis is important to clearly define the project scope (boundaries) and to identify key functionalities of the future system with the customer to avoid problems like forgotten key functionalities or requirements creep.
- Software Component Identification is a key stone of a software project. Failure to describe a design architecture that will incorporate all the requirements is a recipe for disaster. The customer will not finalize the payment if the design doesn’t answer all his requirements.
- The Software Construction is a key stone for programmers that will feel confidence enough to produce components a systematic approach that can be useful for constructing complex components.
- The Software Integration and Tests allows executing different types of tests and identifying issues that must be corrected by the software development team. The different types of tests are executed in different points of time.
- The Product Delivery conducts ongoing activities, there should be no surprise, no delays to obtain acceptance of deliverables. Otherwise, the customer will not finalize the payments to the VSE.

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: set of interrelated or interacting activities which transform inputs into outputs [ISO/IEC 12207].

Activity: a set of cohesive tasks of a process [ISO/IEC 12207].

Task: required, recommended, or permissible action, intended to contribute to the achievement of one or more outcomes of a process [ISO/IEC 12207].

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

Step: In a deployment package, a task is decomposed in a sequence of steps.

Role: a defined function to be performed by a project team member, such as testing, filing, inspecting, coding. [ISO/IEC 24765]

Product: piece of information or deliverable that can be produced (not mandatory) by one or several tasks. *(e. g. design document, source code)*.

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

Agreement: The definition of terms and conditions under which a working relationship will be conducted. [ISO/IEC 12207]

Baseline: a specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures. [ISO/IEC 12207]

Component: Set of functional services in the software, which, when implemented, represents a well-defined set of functions and is distinguishable by a unique name [ISO/IEC 29881:2008]

Defect: A problem which, if not corrected, could cause an application to either fail or to produce incorrect results [ISO/IEC 20926].

***Delivery**: 1. release of a system or component to its customer or intended user.* [ISO/IEC 24765]

Implementation environment: hardware, software, and protocols needed to carry out the implementation.

Non Functional Requirement: a software requirement that describes not what the software will do but how the software will do it. ISO/IEC 24765, Systems and Software Engineering Vocabulary. Syn. design constraints, non-functional requirement. See also: functional requirement. NOTE for example, software performance requirements, software external interface requirements, software design constraints, and software quality attributes. Non functional requirements are sometimes difficult to test, so they are usually evaluated subjectively. [ISO/IEC24765]

Product: the result of a process. [ISO/IEC 12207]

Release: a particular version of a configuration item that is made available for a specific purpose (for example, test release). [ISO/IEC 12207]

Requirement: *1.* a statement that identifies what a product or process must accomplish to produce required behaviour and/or results. *IEEE 1220-2005 IEEE Standard for the Application and Management of the Systems Engineering Process*. 3.1.16. *2.* a system or software requirement that specifies a function that a system/software system or system/software component must be capable of performing. *ISO/IEC 24765, Systems and Software Engineering Vocabulary. 3.* a requirement that specifies a function that a system or system component must be able to perform. [ISO/IEC24765]

Requirements analysis: The process of studying user needs to arrive at a definition of system, hardware, or software requirements. [ISO/IEC 24765]

Requirements document: a document containing any combination of recommendations, requirements or regulations to be met by a software package. [ISO/IEC 24765]

Software Testing Environment: supported software test environment which satisfies the testing requirements and in which changes must be controlled to allow regression if necessary.

Work Breakdown Structure (WBS): a deliverable-oriented hierarchical decomposition of the work to be executed by the project team to accomplish the project objectives and create the required deliverables. It organizes and defines the total scope of the project. [PMI 2008]

3. Relationships with ISO/IEC 29110

This deployment package covers the activities related to Project Management of the ISO Technical Report ISO/IEC 29110 Part 5-1-1 for Very Small Entities (VSEs) – Generic Profile Group: Entry Profile [ISO/IEC29110].

The Guide provides Project Management and Software Implementation processes which integrate practices based on the selection of ISO/IEC 12207- Systems and Software Engineering —Software Life Cycle Processes:2008 and ISO/IEC 15289 Systems and Software Engineering – Software Life Cycle Process – guidelines for the content of software life cycle process information products (documentation):2006 standards elements.

The purpose of the Project Management process is to establish and carry out in a systematic way the tasks of the software implementation project, which allows complying with the project's objectives in the expected quality, time and cost.

The purpose of the Software Implementation process is the systematic performance of the analysis, design, construction, integration and tests activities for new or modified software products according to the specified requirements. Another Deployment Package describes the project management process.

Both processes are interrelated (see Figure 1).

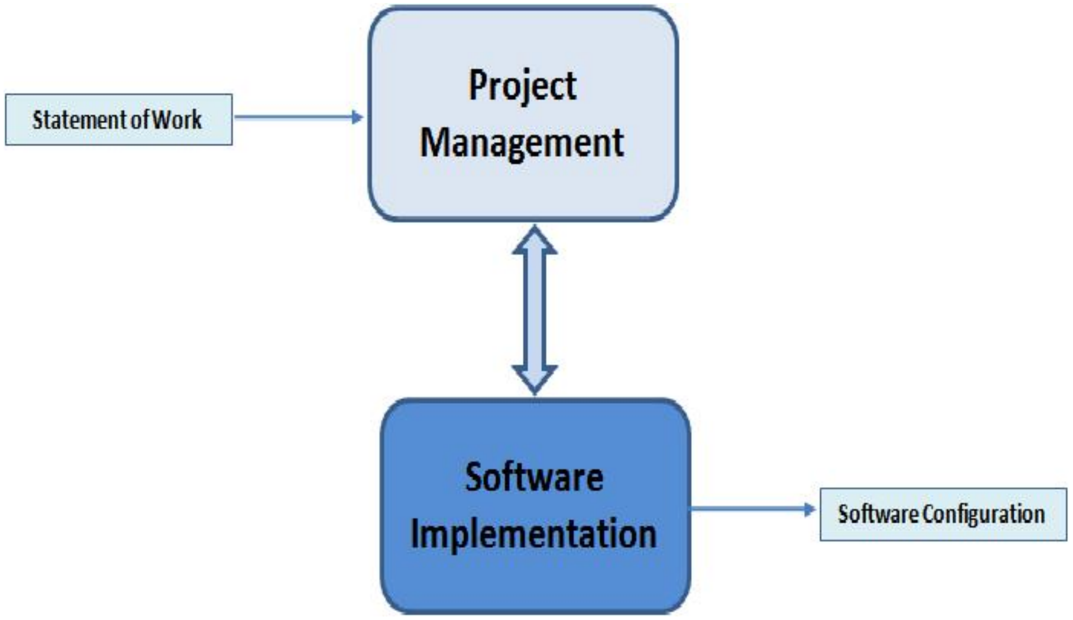

Figure 1 — Entry profile guide processes
(Software Implementation is explained in section 4)

4. Description of Processes, Activities, Tasks, Steps, Roles and Products

The following diagram shows the flow of information between the Software Implementation Process activities including the most relevant work products and their relationship.

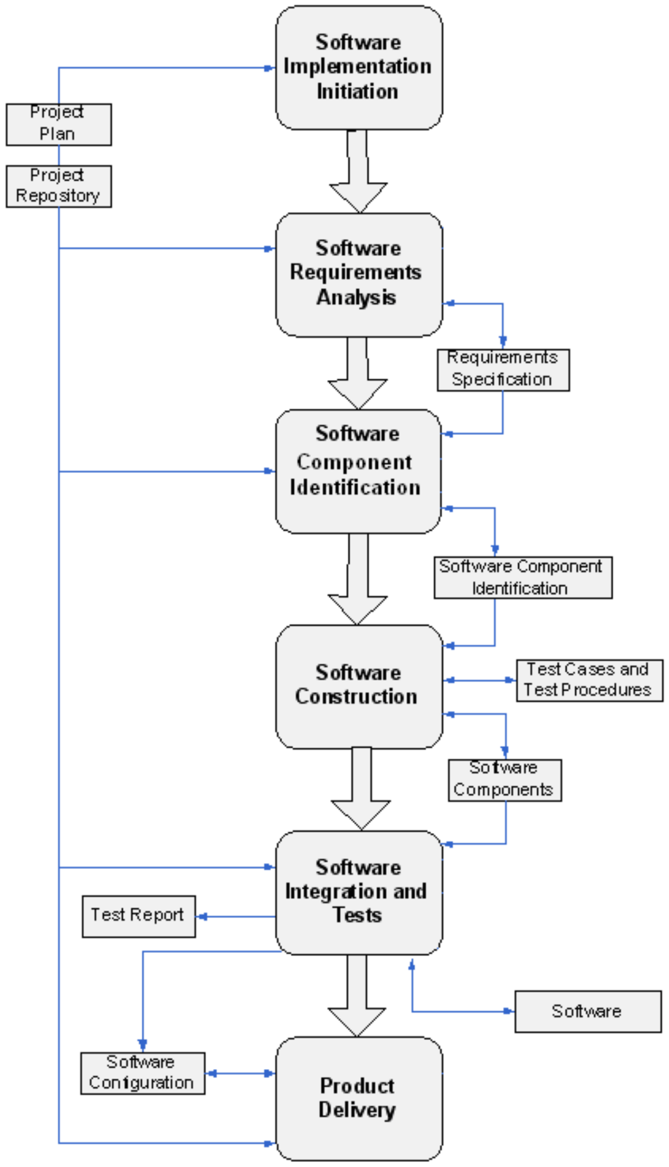
Software Implementation Initiation
Project Plan
Project Repository
Software Requirements Analysis
Requirements Specification
Software Component Identification
Software Component Identification
Software Construction
Test Cases and Test Procedures
Software Components
Software Integration and Tests
Test Report
Software
Software Configuration
Product Delivery

Figure 2 — Software Implementation process diagram (ISO/IEC 29110)

4.1. SI Activities

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

The Software Implementation Process has the following activities:

— SI.1 Software Implementation Initiation

— SI.2 Software Requirements Analysis

— SI.3 Software Component Identification

— SI.4 Software Construction

— SI.5 Software Integration and Tests

— SI.6 Product Delivery

4.1.1. Activity: SI.1 Software Implementation Initiation

The Software Implementation Initiation activity ensures that the *Project Plan* established in Project Planning activity is committed to by the Work Team. The activity provides:

— Review of the *Project Plan* by the Work Team to determine task assignment.

— An implementation environment established.

Task List	Input Products	Output Products	Role
SI.1.1 Review the current Project Plan with the Work Team members in order to achieve a common understanding and get their engagement with the project.	*Project Plan*	*Project Plan[reviewed]*	PM WT
SI.1.2 Set or update the implementation environment.	Project Plan		WT

Software implementation Initiation

Objectives:	To understand the Project Plan by all the members of the work team and to identify and set up the elements of the implementation environment.
Rationale:	This allows prepare the team work for the activities and to have all the necessary tools to accomplish the work.

Roles:	Project Manager
	Work Team
Artifacts:	Project Plan
Steps:	Step 1: Review the current Project Plan with the Work Team members
	Step 2: Set the implementation environment.
Step Description:	***Step 1: Review the current Project Plan with the Work Team members*** The project manager should hold a meeting with all participants or work team to ensure the full understanding of the project and its objectives. The project manager should be sure that every team member has been summoned to the meeting. At the end of the meeting the project manager should check that all the work team understood and accept their engagement at project.
	Step 2: Set the implementation environment. During this step, conditions of the environment over which the implementation will be executed, readiness and availability of components and data that will be used for this execution are prepared. The objective is to define a controlled and independent environment for the implementation.

4.1.2. Activity: SI.2 Software requirements analysis

The Software Requirements Analysis activity analyzes the agreed customer requirements and establishes the validated project software requirements. The activity provides:

- Work Team review of the *Project Plan* to determine task assignment.
- Elicitation, analysis and specification of customer’s requirements
- Agreement on the customer requirements.
- Verification and validation of requirements.

Task List	Input Products	Output Products	Role
SI.2.1 Assign tasks to the Work Team members in accordance with their role, based on the current *Project Plan.*	*Project Plan*		PM WT

Task List	Input Products	Output Products	Role
SI.2.2 Document or update the *Requirements Specification.* Identify and consult information sources (customer, users, previous systems, documents, etc.) in order to get new requirements. Gather and analyze the identified requirements to determinate the scope and feasibility. Verify the correctness and testability of the *Requirements Specification* and its consistency with the *Product Description*. Generate or update the *Requirements Specification.*	*Project Plan (Product Description)*	*Requirements Specification*	*WT* *CUS*
SI.2.4 Validate and obtain approval of the *Requirements Specification* Validate that *Requirements Specification* satisfies needs and agreed upon expectations, including the user interface usability.	*Requirements Specification*	*Requirements Specification [validated]*	*CUS*

Software Requirements Analysis

Objectives:	The objective of this activity is to clearly define the scope of the project and identify the key requirements of the system.
Rationale:	It is important to clearly define the project scope (boundaries) and to identify key functionalities of the future system with the customer to avoid problems like forgotten key functionalities or requirements creep.
Roles:	Work Team
	Customer
Artifacts:	Requirements Document
Steps:	Step 1. Collect information about the application domain (e.g. finance, medical)
	Step 2. Identify project’s scope
	Step 3. Identify and capture requirements
	Step 4. Structure and prioritize requirements

Step Description:	***Step 1. Collect information about the domain*** During this Step, the work team captures the key concepts of the business domain of the customer. The customer assists the work team by giving him all the information (existing documentation or explanation) that will facilitate this understanding. Key concepts are listed in a glossary section in the *Software Requirements Specification Document outline* document. ***Step 2. Identify project's scope*** Software analyst, helped by the person in charge of the contractual aspects of the project (sales manager) clearly identifies main functionalities that are included in the project scope. *Tip*: Identifying functionalities that are OUT of scope is also very valuable to clarify differences of understanding with your customers. ***Step 3. Identify and capture requirements*** Having in mind key concepts related to the customer business domain, analyst can start requirements identification. None of the situations in IT projects are identical. In some cases, most of the requirements are already identified in a document (call for tender in case of fixed priced projects). However, in most of the cases, key requirements are just (orally) mentioned by the customer. Analyst must identify and list the key requirements of the system to be built. During this Step, analyst should not start detailing identified requirements. The main goal is to gain a comprehensive view of the system requirements. ***Step 4. Structure and prioritize requirements***: Using requirements identified in the previous Step, the analyst has to organise and structure identified requirements accordingly (e.g. by business processes or by system functions). A priority must be identified by the customer for system key functionalities. Priorities can be stated like • '*High*' – a functionality that *shall be* implemented • 'Medium' - a functionality that *should be* implemented • 'Low' - a functionality that *could be* implemented The output of this Step is a list of requirements that are organized in the *Requirements Document.*

4.1.3. Activity: SI.3 Software components identification

The Software Component Identification activity transforms the software requirements to the architecture of system software components. The activity provides:

- Work Team review of the *Project Plan* to determine task assignment.
- Identify software components and associated interfaces.

Task List	Input Products	Output Products	Role
SI.3.1 Assign tasks to the Work Team members related to their role according to the current *Project Plan.*	*Project Plan*		PM WT
SI.3.2 Understand Requirements Specifications.	Requirements Specification		WT
SI.3.3 Document or update the Software Components Identification. Analyze the Requirements Specification to generate the components, its arrangement in subsystems and software components defining the internal and external interfaces. Provide the detail of software components and their interfaces to allow the construction in an evident way.	Requirements Specification	*Software Component Identification*	WT

Software components identification

Objectives:	To identify the software components that will answer the requirements asked by the customer, that will be given the ability to be tested before being released and to verify that every requirements is fulfilled.
Rationale:	Software components identification is a key stone of a software project. Failure to describe a design architecture that will incorporate all the requirements is a recipe for disaster. The customer will not finalize the payment if the design doesn't answer all his requirements.
Roles:	Work Team
Products:	Requirement specifications
	Software Configuration
Artifacts:	UML or BPMN Diagrams
Steps:	Step 1. Understand Requirements Specification
	Step 2. Document or update the Software Components

Step Description:	***Step 1. Understand Requirements Specification*** ○ Examine each requirements and be sure they are understood and grouped : • Functional requirements in logical groups. • Non-functional requirements in groups. ○ Verify groups and check if all requirements are understood. ○ If needed, update the Requirements Specifications to add necessary clarification. • Store updated document in repository ***Step 2. Document or update the Software components identification*** ○ Analyze the *Requirements Specification* to generate the components, its arrangement in subsystems and components defining the internal and external interfaces. ○ Describe in detail, the appearance and the behaviour of the interface, based on the *Requirements Specification* in a way that resources for its implementation can be foreseen. ○ Provide the detail of *Components* and their interfaces to allow the construction in an evident way.

4.1.4. Activity: SI.4 Software construction

The Software Construction activity develops the software code and data from the Software Components identified in the SI.3. The activity provides:

— Work Team review of the *Project Plan* to determine task assignment.

— Understand the identified Software Components.

— *Test Cases and Test Procedures* for unit and integration testing.

— Coded *Software Components* and applied unit tests.

Task List	Input Products	Output Products	Role
SI.4.1 Assign tasks to the Work Team members related to their role, according to the current *Project Plan.*	*Project Plan*		PM WT
SI.4.2 Understand the identified	Software Components identification		WT

Task List	Input Products	Output Products	Role
Software Components			
SI.4.3 Construct or update Software Components.	Software Components identification	Software Components	WT
SI.4.4 Establish or update *Test Cases and Test Procedures* for unit and integration testing based on *Requirements Specification* and *Software Component Identification.* Customer provides testing data, if needed.	*Requirements Specification [validated]* *Software Component Identification*	*Test Cases and Test Procedures*	WT
SI.4.5 Test the Software Components. Correct the defects found until successful unit test is achieved.	*Test Cases and Test Procedures* *Software Components*	*Software Components[unit tested]*	WT

Software construction

Objective:	Produce the Software Components as identified.
Rationale:	Programmers may feel confidence enough to produce components without a systematic approach, however some of them can find it useful for constructing complex components. There are several approaches to produce components, here we use the Pseudocode approach which is one of the most widespread accepted. Steps were adapted from Code Complete (see reference).
Roles:	Work Team
Products:	Software Components
Artifacts:	Software Components
Steps:	Step 1. Design the Component(s)

	Step 2. Code the Component
	Step 3. Verify the Component
Step Description:	***Step 1. Design the Component(s)*** a) **Verify the contribution of the Component.** Understand the contribution of the Component to the final product. b) **Define the problem the Component will solve.** State the problem the component will solve in enough detail to allow its creation. If the information is not enough the programmer should complete it with the support of the Designer. c) **Research functionality available in the standard libraries**. Find out whether some or all of the component's functionality might already be available in the library code of the language, platform, or tools you're using. d) **Write the pseudocode.** Start with the general and work toward something more specific. The most general part of a Component is a header comment describing what the component is supposed to do, so first write a concise statement of the purpose of the Component, after that decompose the statements in several low level statements, always taking care of completeness. After creating the mainstream of the Component include **error handling,** managing all the things that could possibly go wrong in the component, like: bad input values, invalid values returned from other routines, and so on. e) **Design the component's data.** Define key data types f) **Check the pseudocode.** Review the pseudocode, if you are not very confident about it, explain it to someone else (could be the Designer), this explanation will help to clarify your ideas and may improve the pseudocode. ***Step 2. Code the component***(s) a) **Write the component declaration** and turn the original header comment into a programming-language comment. b) **Turn the pseudocode into high-level comments** c) **Fill in the code below each comment** Fill in the code below each line of pseudocode comment. Each pseudocode comment describes a block or paragraph of code. d) **Check whether code should be further factored**

	In some cases, you'll have lot of code below one of the initial lines of pseudocode. In this case, you should consider creating a subcomponent. ***Step 3. Verify the component*** This step consists of a code review performed by the programmer, a compilation and debugging. **a) Review the code** - Check the program against the requirements identified to ensure that all required program functions are implemented. - Follow the code review checklist, provided in section 7, to find all the defects in the program. - In doing the review, mark the defects on the source code - Following the code review, correct all the defects. - After completing the corrections, produce a source program listing for the corrected program. - Review all the corrections to ensure that they are correct. - Correct any remaining defects. **b) Compile the code** Let the computer check for undeclared variables, naming conflicts, and so on, you may have done this before the Code Review also. **c) Step through the code with a debugger** Once the routine compiles, put it into the debugger and step through each line of code. Make sure each line executes as you expect it to. You may do this after unit test focusing in finding the source of a defect.

4.1.5 Activity: SI.5 Software Integration and Tests

The Software Integration and Test activity ensures that the integrated software components satisfy the software requirements. The activity provides:

— Work Team review of the *Project Plan* to determine task assignment.

— Understanding of *Test Cases and Procedures* and the integration environment.

— Integrated *Software Components*, corrected defects and documented results.

Task	Input products	Output products	Roles
SI.5.1 Assign tasks to the work team members related to their role, according to the	*Project Plan*		PM WT

current *Project Plan.*			
SI.5.2 Understand *Test Cases and Test Procedures.* Set or update the testing environment.	*Test Cases and Test Procedures*		WT
SI.5.3 Integrates the *Software* using *Software Components* and updates Test Cases and Test Procedures for integration testing, as needed.	*Software Components* *Test Cases and Test Procedures*	*Software* *Test Cases and Test Procedures [updated]*	WT
SI.5.4 Perform *Software* tests using *Test Cases and Test Procedures* for integration and document results in *Test Report*.	*Software* *Test Cases and Test Procedures*	*Software [tested]* *Test Report*	WT
SI.5.5 Correct the defects found until successful test is achieved.	*Software, Test Report.* *Test Cases and Test Procedures*	*Software [corrected]* *Test Report [defects eliminated]*	WT
SI.5.6 Incorporate the Requirements Specification and Software to the Software Configuration.	*Requirements Specification* *Software*	*Software Configuration* *-Requirements Specification* *- Software*	WT

Software Integration and Tests

Objectives:	To identify software product stability through obtained results, exposing the product to tests.
Rationale:	This allows executing different types of tests and identifying issues that must be corrected by the software development team. The different types of tests are executed in different points of time.
Roles:	Customer
	Analyst
Artifacts:	Test Report

Steps:	Step 1: Prepare the software testing environment
	Step 2: Execute testing iterations
	Step 3: Execute regression tests
	Step 4: Close testing procedure
	Step 5: Document results in *Test Report*
Step Description:	***Step 1: Prepare the software testing environment*** During this step, conditions of the environment over which tests will be executed; readiness and availability of components (corrected or baselined) to be tested and data that will be used for this execution are prepared. The objective is to define a controlled and independent environment for the tests. Preparing the testing environment includes: Specifying machine configuration, operating system, browser and TCP/IP configuration when it applies; specifying system software, data base engine and testing support engine. Regarding testing data, when the application is completely new, tests are executed in the sequence necessary to manually load initial data; otherwise, data from a client in operation, i.e., reproduction data, will be reused. ***Step 2: Execute testing iterations*** Before execution it is necessary to guarantee that assigned analyst understands the tests. A testing iteration corresponds to the execution of tests. Testing iteration is done once the development team has executed unit testing. According to application stability, the analyst may decide, in each iteration, executing all identified testing requirements or just a representative subset. The purpose is accomplishing an efficient benefit relationship between the invested effort in the execution and the number of issues. Through the execution of tests, the analyst finds deviations to the expected results, catalogued as product 'Defects'. Product defects identified are typified and recorded in the testing follow-up and control report that may also be a buff tracking tool. For Defect classification, the following can be taken into account: Blockers: This type of issue stops a program/component operation of makes it yield results that prevent continuing operation. Functional: It occurs when a program is executed and its results do not correspond to the expected result. Presentation: Issues related to product presentation. These must adjust to the defined standards and the grammatical and orthographic rules of the language in which the program is showed.

	Step 3: Execute regression tests The objective is to make sure that new or modified code (to provide solution to issues identified during tests) complies with the specified requisites and that non modified code has not been affected by the maintenance activity. Regression tests initiate their execution as soon as the technical group delivers the issue solution or adds new characteristics to the software and these are applied in the testing controlled environment. To achieve an efficient procedure, it is possible to combine the regression test with a testing iteration. ***Step 4: Close testing procedure*** Once defined testing iterations have been completed and identified defects have been solved, an analysis to demonstrate that there is a *sustainable tendency to decrease* in the number of found defect is carried out. The closure of testing procedure is done elaborating a closure report, which describes the executed procedure and shows conclusions with corresponding recommendations for process and product improvement. These recommendations make part of the lessons learned base of the VSE. The development team decides whether or not to release the software product, according to results obtained in the tests. ***Step 5: Document results in Test Report*** The analyst or programmer must prepare a progress test report for each testing iteration carried out as a minimum. The report must contain product current condition. The report must be reviewed by all members of the project team, and they must establish commitments for the new testing iteration.

4.1.6. Activity: SI.6 Product delivery

The Product Delivery activity provides the integrated software product to the Customer. The activity provides:

— Work Team review of the *Project Plan* to determine task assignment.

— Delivery of the software product and applicable documentation in accordance with the Project Plan.

Task List	Input Products	Output Products	Role

Task List	Input Products	Output Products	Role
SI.6.1 Assign tasks to the work team members related to their role, according to the current *Project Plan.*	*Project Plan*		PM WT
SI.6.2 Understand *Software Configuration*.	*Software Configuration*		WT
SI.6.3 Perform delivery according to the Project Plan.	*Project Plan,* *Software Configuration*	*Software Configuration [delivered]*	*WT*

PRODUCT DELIVERY

Objectives:	To conduct ongoing delivery activities such that, at the end of the project, every deliverable is available and meet the acceptance criteria defined in the delivery instructions.
Rationale:	By conducting ongoing activities, there should be no surprise, no delays to obtain acceptance of deliverables. Otherwise, the customer will not finalize the payments to the VSE.
Roles:	Work Team
Products:	Software Configuration
	Acceptance Record Form
Artifacts:	Project Plan
	Approved Delivery Instructions Form
	Acceptance Record Form
Steps:	Step 1. Understand Software Configuration.
	Step 2. Perform delivery according to Delivery Instructions.
Step Description:	***Step 1. Understand Software Configuration.*** • Obtain, from the project plan, the delivery conditions. • Build/Obtain the product baseline o Analyst prepares the deliverables. o Analyst establishes the baseline for the product configuration including environment relevant, manual, designed document and configured product. ***Step 2.Perform delivery according to Delivery Instructions.*** • Verify that each software component meets the acceptance

	criteria. • Update the Acceptance Record Form • Plan a meeting with the customer • Obtain approval of Acceptance Record Form from the customer • The customer will sign the Acceptance record Form • Give a copy of the Acceptance Record Form to the customer • Store the Acceptance record Form in the repository

4.2 Role Description

This is an alphabetical list of the roles, its abbreviations and suggested competencies description. This list is showed as a four-column table for presentation purpose only.

	Role	**Abbreviation**	**Competency**
1.	Customer	CUS	Knowledge of the Customer processes and ability to explain the Customer requirements. The Customer (representative) must have the authority to approve the requirements and their changes. The Customer includes user representatives in order to ensure that the operational environment is addressed. Knowledge and experience in the application domain.
2.	Project Manager	PM	Leadership capability with experience making decisions, planning, personnel management, delegation and supervision, finances and software development.
3.	Work Team	WT	Knowledge and experience according to their roles on the project.

4.3. Product description

This is an alphabetical list of the input, output and internal process products, its descriptions, possible states and the source of the product. The source can be another process or an external entity to the project, such as the Customer. This list is showed as a four-column table for presentation purpose only. Product items in the following tables are based on ISO/IEC15289 Information Items with some exceptions.

	Name	**Description**	**Source**
1.	*Acceptance Record*	Documents the customer acceptance of the deliverables of the project. It may have the following characteristics: - Record of the receipt of the delivery - Identifies the date received - Identifies the delivered elements - Record of customer verification of the deliverables according to the Customer agreement. - Identifies any open issues (if applicable) - Signed by receiving Customer	Project Management
2.	*Change Request*	Identifies a software, or documentation problem or desired improvement, and requests modifications. It may have the following characteristics: - Identifies purpose of change - Identifies request status (new, accepted, rejected) - Identifies requester contact information - Impacted system(s) - Impact to operations of existing system(s) defined - Impact to associated documentation defined - Criticality of the request, date needed by The applicable statuses are: accepted and tracked.	Software Implementation Customer Project Management
3.	*Meeting Record*	Records the agreements established with Customer and/or Work Team. It may have the following characteristics: - Purpose of meeting - Attendees - Date, place held - Reference to previous minutes - Identifies issues raised - Any open issues	Project Management

	Name	Description	Source
		- Agreements - Next meeting, if any. The applicable status is: updated.	
4.	Progress Status Record	Records the status of the project against the Project Plan. It may have the following characteristics: - Status of actual tasks against planned tasks - Status of actual results against established objectives / goals - Status of actual resource allocation against planned resources - Status of actual cost against budget estimates - Status of actual time against planned schedule - Status of actual risk against previously identified - Record of any deviations from planned tasks and reason why. The applicable status is: evaluated.	Project Management
5.	*Project Plan*	Presents how the project processes and activities will be executed to assure the project's successful completion, and the quality of the deliverable products. It Includes the following elements which may have the characteristics as follows: - *Product Description* ○ Purpose ○ General Customer requirements - *Scope* description of what is included and what is not - *Deliverables* - list of products to be delivered to Customer - *Tasks, including* verification, validation and reviews with Customer and Work Team, to assure the quality of work products. Tasks may be represented as a Work Breakdown Structure (WBS). - *Relationship and Dependence of the Tasks* - *Estimated Duration* of tasks - *Resources* (humans, materials, standards, equipment and tools)~~,~~ and the schedule when the resources are needed.	Project Management

	Name	Description	Source
		- Composition of *Work Team* - *Schedule of the Project Tasks,* the expected start and completion date, for each task. - *Estimated Effort and Cost* - Identification of Project Risks The applicable statuses are: accepted	
6.	*Project Repository*	Electronic container to store project work products and deliveries. It may have the following characteristics: - Stores project work products - Stores released deliverables products - Storage and retrieval capabilities - Ability to browse content - Listing of contents with description of attributes - Sharing and transfer of work products between work team. - Effective controls over access - Maintain work products descriptions - Recovery of archive versions of work products - Ability to report work products status - Changes to work products are tracked to *Change Requests* The applicable statuses are: updated.	Project Management
7.	*Requirements Specification*	Identifies the software requirements. It may have the following characteristics: - Introduction –general description of software and its use within the scope of the customer business; - Requirements description: - Functionality – established needs to be satisfied by the software 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 software easily so the user be able to perform his/her tasks efficiently including the	Software Implementation

	Name	Description	Source
		interface exemplar description; - External interfaces – definition of interfaces with other software or hardware; Each requirement is identified, unique and it is verifiable or can be assessed. The applicable statuses are: verified, and validated.	
8.	*Software*	Software item (software source and executable code) for a Customer, constituted by a collection of integrated *Software Components*. The applicable statuses are: tested	Software Implementation
9.	*Software Component*	A set of related code units. The applicable statuses are: unit tested	Software Implementation
10.	*Software Configuration*	A uniquely identified and consistent set of software products including: - *Requirements Specification* - *Software* The applicable statuses are: delivered and accepted.	Software Implementation
11.	*Software Component Identification*	Textual and graphical information on the software structure. This structure may include the following parts: Describes the overall *Software* structure: - Identifies the required *Software Components* - Identifies the relationship between *Software Components*	Software Implementation
12.	*Statement of Work*	Description of work to be done related to software development. It may Include: - Product Description o Purpose o General Customer requirements - Scope description of what is included and what is not - Deliverables list of products to be delivered to Customer The applicable status is: reviewed.	Customer
13.	*Test Cases and Test*	Elements needed to test code. Test Case may include: - Identifies the test case	Software Implementation

	Name	Description	Source
	Procedures	- Test items - Input specifications - Output specifications - Environmental needs - Special procedural requirements - Interface dependencies Test Procedures may include: - Identifies: test name, test description and test completion date - Identifies potential implementation issues - Identifies the person who completed the test procedure - Identifies prerequisites - Identifies procedure steps including the step number, the required action by the tester and the expected results	
14.	*Test Report*	Documents the tests execution. It may include: - A summary of each defect - Identifies the tester who found each defect - Identifies the affected function(s) for each defect - Identifies the date when each defect originated - Identifies the date when each defect was resolved - Identifies the person who resolved each defect	Software Implementation

4.4. Artifact description

Artifacts	**Definition**
Acceptance document	Document establishing the customer acceptance of the deliverables established on the project.
Project Description	A high level description of the project to include; Scope; Objectives and major Deliverables.
Project Plan	A statement of how and when a project's objectives are to be achieved, by showing the major products, milestones, activities and resources required on the project.
Requirements Document	Document in which all identified requirements are centralized.
Software	A consistent set of software products which include: • Requirements Specification

	• Software Components • Software (unit, product, item) • Test and Incident Reports • Operational Manual • User Manual
Software Component	A set of related code units. The applicable statuses are: unit tested, corrected and baselined.
Software Configuration	A uniquely identified and consistent set of software products including: - *Requirements Specification* - *Software Design* - *Software Components* - *Software* - *Test Report* - *Product Operation Guide* - *Software User Documentation* - *Maintenance Documentation*
Test Report	Record of testing execution results in a specific environment.
UML Diagrams	These diagrams will facilitate the graphical representation of the design by using Unified Modelling Language. UML can provide an easy and standard way to express data, processes or architecture.

5. Template & Tools

The templates are provided as examples, they should be customized for your project.

5.1. Software Implementation Initiation Template

Table of Contents Template – Meeting notification

1. Meeting Name (includes project name) ……………….
2. Meeting priority ……………….
3. Addressees ……………….
4. Meeting objective ……………….
5. Message content ……………….
6. Date, place and time ……………….
7. Others (Ex. food included) ……………….

Assistance control list template

No.	*Work team member name*	*Role*	*Assistance*		*Signature*
			Yes	*Not*	
Comments					

5.2. Software requirements analysis Template

Table of Requirements

To be used in an Excel sheet structured, for example, as:

ID	*Requirement*	*Description*	*Priority*		
			High	*Medium*	*Low*

Each requirement can be defined as a table. The structure of the table is as follows:

Project	*<Project identification>*
Date	<Date of last change>
Requirement	<Identification, Text of the requirement>
Description	<Detailed description of the requirement>
Rationale	<Why this requirement is important>

Priority	<Priority rate of the requirement>
Dependency	<Relation with others requirements>
Comment	<additional remarks>

5.3 Software Component Identification Template

Links to diagram standards.

Template	Source
UML	http://www.uml.org
BPMN	http://www.bpmn.org

To be used in an Excel sheet structured, for example, as:
Table of Contents Template – Components Identification

Project Identification (Name, date)
1. Component 1.
   1.1. Identification
   1.2. Description
   1.3. Sub-components
   1.4. Requirements
2. Component 2.
   2.1. Identification
   2.2. Description
   2.3. Sub-components
   2.4. Requirements
3. …

5.4 Software Construction

Links to coding standards templates

Template	Source
General coding Standard template	http://www.construx.com

Source Code template C / C++	http://www.construx.com
Source Code template Java	http://www.construx.com
Coding Standards Java	http://www.ontko.com/java/java_coding_standards.html
Source Code template Visual Basic	http://www.construx.com
Source Code template XML	http://www.construx.com

Links to a User Interface Specification template

Template	**Source**
User Interface Specification	http://www.hcirn.com/tutor/docs/uitempl.php

5.5 Test Specifications Template

To be used in an Excel sheet structured, for example, as:

Application type	
Operation Mode	
Topics	
Target Group	
Types of Media to be used	

Client Machine	Software Characteristics	Version
Operating System		
Browser		
Browser Requirements		
Communication Protocol		
	Hardware Characteristics	**Capacity**
Processor		
Hard Disk		
RAM Memory		
Screen Resolution		
Multimedia	(Applicable only to Multimedia)	
CD ROM Speed		
Video Card		
Sound Card		
3D Accelerator		
	Additional Hardware Characteristics of the Client	**Capacity**
Printer		
Barcode Scanner		

Client Machine	Web Server Software Characteristics	Version
Operating System		
Web Server		
Development Language		
Language Requirements		
	Database Server Software Characteristics	**Version**
Operating System		
Database		
	Web Server Hardware Characteristics	**Capacity**
Processor		
Hard Disk		
RAM Memory		

Table of Contents Template – Test Report

c. Minor failures	………………..
4. Components integration	
a. Test log	
b. Test documentation	
c. Test data sheets	
d. Analysis and conclusions	………………..
5. Raw test data file	………………..
6. General conclusions	……….…….

5.6 Delivery Product Template

Delivery Instructions Form

Delivery Instructions Form

Project Identification or Name of Customer: ______________________

Prepared by (Name Initial): ___________________________

Date (yyyy-mm-dd): _________________

Identification of Deliverables (i.e., hardware, software, documentation etc.)**:**

Delivery requirements:

Sequential ordering of tasks to be performed:

Applicable releases:

Acceptance Criteria:

Acceptance Criteria	**Date Criteria is met (yyyy-mm-dd)**
1	
2	
3	

software components	**Version information**

Backup and recovery procedures:

Approved by:

______________________ ________________________________
Project Manager **Customer or Customer Representative**

Date (yyyy-mm-dd):________________

Acceptance Record Form

Acceptance Record Form

Project Identification:

Prepared by (Name Initial):
Date (yyyy-mm-dd):

Acceptance Criteria:

Acceptance Criteria	Date Criteria is met (yyyy-mm-dd)
1	
2	
3	
4	
5	

Acceptance Signatures:

Software components	Version information	Date accepted by Customer	Signature of Customer

Signatures:

________________________ ________________________________
Project Manager **Customer or Customer Representative**

Date (yyyy-mm-dd)**:**____________________

Tools

This section gives information about tool sources which could help to support some of the software implementation process activities.

Software integration and tests

To ensure control of product issues, it is necessary to implement a system, preferably a web system, that allows doing a follow-up of reported issues about a product, and processing requests associated to them.

As key features, this system must allow:

- Automatically assigning issues
- Attaching documents
- Generating automatic notifications via e-mail
- Controlling changes
- Generating simple and interactive queries
- Issue classification by: Type, Cause, Severity.

There are several open source testing tools on internet, here is a web site offering a few of them:

http://www.opensourcetesting.org/functional.php

Product Delivery

- Version Control tools such as CVS (Concurrent Versions System) or SVN (version management).
- Repositories available from Open Source: e.g. Google Doc
- Configuration Management tools

6. Example of Activity lifecycle

Disclaimer: *This section provides some graphical representations of examples of testing software practices lifecycle. These examples are provided to help the reader in the implementation of his own testing software lifecycle fitting his IT project's context and constraints.*

Example of Software Implementation Lifecycle

This is an example – use SPEM stencil for Microsoft Visio (http://www.pa.icar.cnr.it/cossentino/FIPAmeth/docs/SPEM.vss) in order to produce such a diagram.

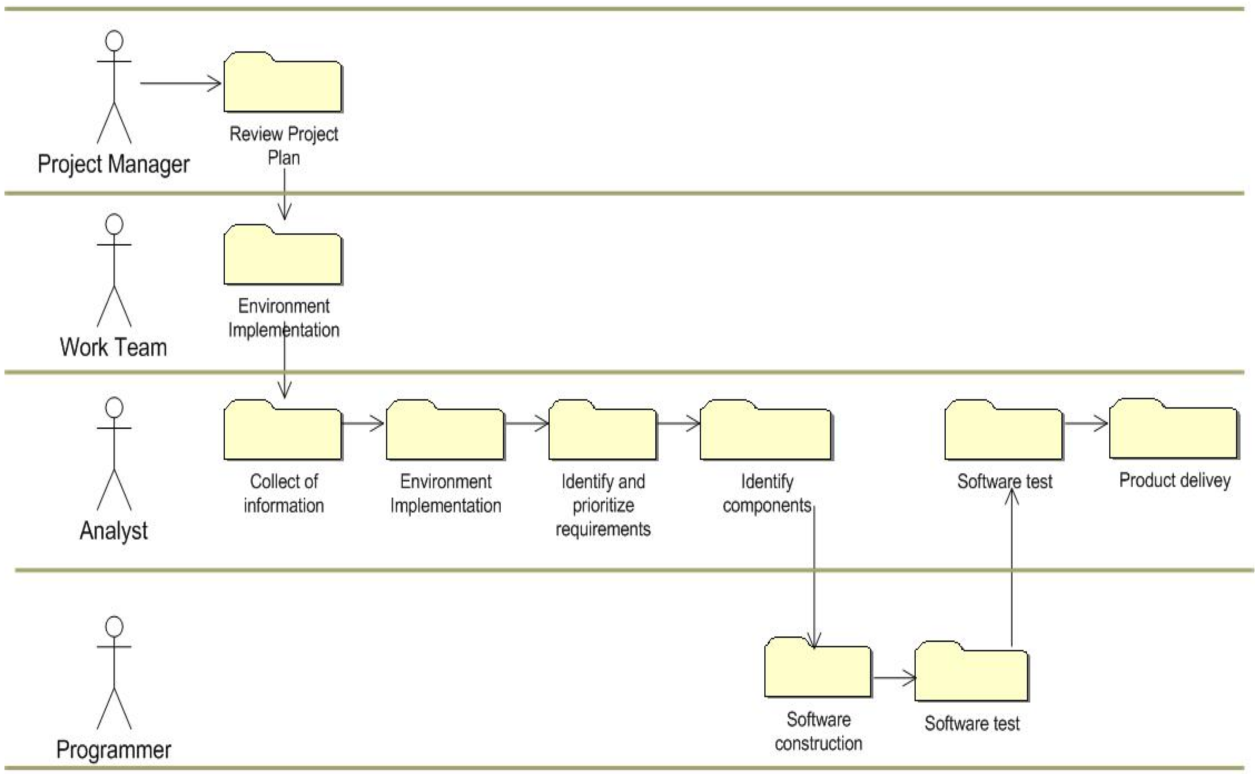

Figure 3 Example of Software Implementation Lifecycle

7. Checklist

7.1 Software Requirements Analysis Checklist

This Requirement checklist is based on [Constr07]

RS 1 Testable	All requirements are verifiable (objectively)
RS 2 Complete	Are the requirements complete?
RS 3 Correct	Requirements must be correct (i.e. reflect exactly customer's requirements)
RS 4 Unique	Requirements must be stated only once
RS 5 Elementary	Requirements must be broken into their most elementary form
RS 6 Scope	Are the requirements in scope?
RS 7 High Level	Requirement must be stated in terms of final need, not perceived means (solutions)
RS 8 Unambiguous	Requirements must be stated in terms that can be interpreted in one way only.

7.2 Software Component Identification Checklist

RS 1 Testable	All components are verifiable (objectively)
RS 2 Complete	Are the components complete?
RS 3 Correct	Components must be correct (i.e. reflect exactly one part of software solution)
RS 4 Unique	Components must be stated only once
RS 5 Elementary	Components must be broken into their most elementary form
RS 6 Linked	Components and Sub-Component are linked
RS 7 Associate	All requirements must be associated to the components.
RS 8 Unambiguous	Components must be stated in terms that can be interpreted in one way only.

7.3 Software Construction Checklist

Code Review Checklist

Note: This checklist can be adapted to apply to the language you are programming in.

Subject	**Description**
Complete	- Verify that all functions in the design are coded and that all necessary functions and procedures have been implemented.
Logic	- Verify that the program flow and all procedure and function logic is consistent with the detailed design.
Loops	- Do a hand simulation of all loops and recursive procedures that were not simulated in the detailed design review or inspection. - Ensure that every loop is properly initiated and terminated. - Check that every loop is executed the correct number of times.
Calls	- Check every function and procedure call to insure that it exactly matches the definition for formats and types.
Declarations	Verify that each variable and parameter - has exactly one declaration - is only used within its declared scope - is spelled correctly wherever used
Initialization	- Check that every variable is initialized.
Limits	- Check all variables, arrays, and indexes to ensure that their use does not exceed declared limits.
Begin-end	- Check all begin-end pairs or equivalents, including cases where nested ifs could be misinterpreted.
Boolean	- Check Boolean conditions
Format	- Check every program line for instruction format, spelling, and punctuation.
Pointers	- Check that all pointers are properly used.
Input-output	- Check all input-output formats.
Spelling	- Check that every variable, parameter, and key work is properly spelled.
Comments	- Ensure that all commenting is accurate and according to standard.

7.4 Software Integration and Tests Checklist

Information should be taken from Construx[1]
http://www.construx.com/Page.aspx?nid=208

7.5 Product Delivery Checklist

Typical Acceptance Criteria for a VSE

- The product supporting media is labelled correctly, showing at a minimum product name, release date, and correct version number.
- Product labelling, including delivery location and product acceptance personnel (if applicable), is accomplished.
- The software generated from the project repository in accordance with the delivery instructions
- The software to be delivered is the latest version of the software in the project repository
- The Version Description Document has been inspected.
- The Version Description Document is included with the supporting media.
- All the tests have been performed successfully
- All errors have been corrected
- All documentations have been updated
- The User's Manual has been inspected.
- The User's Manual is included with the supporting media.
- All topics in the approved Delivery Instructions Forms have been covered
- The Acceptance Record Form has been updated and ready for signature
- The information needed for delivery (e.g. Site address, customer representative) have been verified before delivery
- Your customer has been informed when a delivery will be performed
- The customer informed you that all his preparations for delivery have been completed

Typical Acceptance Criteria for a Customer

- Delivered software components comply with the approved Delivery Instructions
- Delivered software components comply with its requirements

[1] http://www.construx.com

- Expected documentation, test records, design information, training material etc have been delivered
- Delivered software components operate in its intended environment (both technical and organisational) and is acceptable to users
- Acceptance Report Form is reviewed and signed

8. References

Key	Reference
[MCC 04]	Steve McConnell, Code Complete, Second Edition, Redmond, Wa.: Microsoft Press, 2004.
[IEEE 1012-2004]	IEEE 1012-2004 IEEE Standard for Software Verification and Validation
[ISO/IEC 12207]	ISO/IEC 12207:2008 Systems and software engineering – Software life cycle processes.
[ISO/IEC 15289]	ISO/IEC 15289:2006 Systems and software engineering - Content of systems and software life cycle process information products (Documentation)
[ISO/IEC 29110]	ISO/IEC TR 29110-5-1-1, Software Engineering—Lifecycle Profiles for Very Small Entities (VSEs) – Part 5-1-1: Management and Engineering Guide –Generic Profile Group: Entry Profile. Once published by ISO, ISO/IEC TR 29110-5-1-1 will be available at no cost on the following ISO site: http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html
[ISO/IEC 24765]	ISO/IEC 24765 Systems and software engineering vocabulary An electronic version of the glossary is available at: http://pascal.computer.org/sev_display/index.action
[IEEE 1233-1998]	IEEE Guide for Developing System Requirements Specifications
[PMI 2008]	A Guide to the Project Management Body of Knowledge, Project Management Institute, 2008.
[crm.com]	http://searchcrm.techtarget.com/ . Consulted the 10 July 2010.

9. Evaluation Form

Deployment Package - Software Implementation – Entry Profile V 0.2

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?

q *Very Satisfied* q *Satisfied* q *Neither Satisfied nor Dissatisfied* q *Dissatisfied* q *Very Dissatisfied*

2. The sequence in which the topics are discussed, are logical and easy to follow?

q *Very Satisfied* q *Satisfied* q *Neither Satisfied nor Dissatisfied* q *Dissatisfied* q *Very Dissatisfied*

3. How satisfied were you with the APPEARANCE/FORMAT of this deployment package?

q *Very Satisfied* q *Satisfied* q *Neither Satisfied nor Dissatisfied* q *Dissatisfied* q *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?

q *Definitely* q *Probably* q *Not Sure* q *Probably Not* q *Definitely Not*

Optional

- Name: ______________________________________
- e-mail address : ______________________________________

Email this form to: claude.y.laporte@etsmtl.ca