Homework #2: “Process Assessment and Process Improvement Plan”

CPSC 544: Advanced Software Process, Section 50

Instructor: Dr. Chang-Hyun Jo

Team 03: Caliware

Date Submitted: 12/3/2015

|  |  |  |
| --- | --- | --- |
| Timothy Cioffi-Dinkel | Anthony Farina | Joanna Hang |
| timothycioffi@csu.fullerton.edu | afarina@csu.fullerton.edu | johang@fullerton.edu |
| https://lh5.googleusercontent.com/PSroDgY8ZOUkmCV05kKayV-boTdpJSN4FXzGMZ1w6lwE5Br5w20a5pWMewmiTxHAznLEpv6jhuOyAIByZrQkCyPaceyEEcTEmPBQpFilW6976BlGvS6ZX0VZf9cAObv8=s1600 |  | sig.jpg |

|  |  |
| --- | --- |
| Lourdes Lopez | David Sullivan |
| lourdeslopez@csu.fullerton.edu | dsullivan@fullerton.edu |
| Untitled.png | https://lh5.googleusercontent.com/3FGNbflMbPEB_7oxSHs8FcMMr_Lp5KykcuOeEjncWoizP2v-jn12POyZmVra2fZCxjqKTL1FQzXUmPlmarUW9EyxEf-ii570A3UFlycFawZP6PRs9L7I6VX2bGtZIwEE8Ws=s1600 |

Revision History

|  |  |  |
| --- | --- | --- |
| Date | Description | Team Member |
| 10/20/15 | Outline | Joanna Hang |
| 10/20/15 | Steps of Process Improvement Diagram | Anthony Farina |
| 10/20/15 | Objectives | David Sullivan |
| 11/1/15 | Initiate Process Improvement Cycle | David Sullivan |
| 11/4/15 | Mapping ScrumXP to ISO 12207 System Context Processes | David Sullivan, Joanna Hang |
| 11/4/15 | Mapping ScrumXP to ISO 12207 Software Specific Processes | Anthony Farina, Lourdes Lopez |
| 11/10/15 | Gaps to ISO 12207 | David Sullivan, Lourdes Lopez, Joanna Hang, Anthony Farina |
| 11/18/15 | Rating | David Sullivan, Lourdes Lopez, Joanna Hang, Anthony Farina |
| 11/25/15 | Introduction, Business Goals | Joanna Hang |
| 11/27/15 | Capability Determination Chart | Anthony Farina |
| 11/27/15 | Target/Assessed Capability Chart | Anthony Farina |
| 11/25/15 | Initiate Process Improvement | Joanna Hang |
| 11/25/15 | Review Organization’s Improvement Objectives | Anthony Farina |
| 11/27/15 | Action Plan | David Sullivan, Lourdes Lopez, Joanna Hang, Anthony Farina, Timothy Cioffi-Dinkel |

Table of Contents

[Introduction 4](#_Toc436853942)

[Step 1 – Examine organization‘s business goals 6](#_Toc436853943)

[Business goals 6](#_Toc436853944)

[Objectives of Process Improvement 6](#_Toc436853945)

[Step 2 – Initiate process improvement cycle 7](#_Toc436853946)

[Process Improvement Program 7](#_Toc436853947)

[Sponsorship 7](#_Toc436853948)

[Project Management 7](#_Toc436853949)

[Budget for Process Improvement Cycle 7](#_Toc436853950)

[Accountability 8](#_Toc436853951)

[Process Improvement Program Plan 8](#_Toc436853952)

[Background, History, Current Status of Organizational Process Improvement Activities 8](#_Toc436853953)

[Improvement Objectives 8](#_Toc436853954)

[Organizational Scope 9](#_Toc436853955)

[Process Scope 9](#_Toc436853956)

[Process Improvement Lifecycle 9](#_Toc436853957)

[Key Roles and Responsibilities 9](#_Toc436853958)

[Resources 10](#_Toc436853959)

[Appropriate Milestones, Review Points, Mechanisms 10](#_Toc436853960)

[Risks Associated with the Program and Risk Management 11](#_Toc436853961)

[Assessment Activities Performed 11](#_Toc436853962)

[Assessment Outputs 11](#_Toc436853963)

[Step 3 – Assess current capability 12](#_Toc436853964)

[Comparison/Mapping (Scrum & XP to ISO 12207) 12](#_Toc436853965)

[Gap Analysis 49](#_Toc436853966)

[Rating 70](#_Toc436853967)

[Capability Determination 74](#_Toc436853968)

[Target and Assessed Process Profiles 75](#_Toc436853969)

[Step 4 – Develop action plan 76](#_Toc436853970)

[Identify improvement areas 76](#_Toc436853971)

[Analyze assessment strengths and weaknesses 76](#_Toc436853972)

[Review organizational improvement objectives 77](#_Toc436853973)

[Analyze effectiveness measurements 78](#_Toc436853974)

[Define detailed improvement objectives and set targets 79](#_Toc436853975)

[Derive action plan 80](#_Toc436853976)

[Step 5 – Implement improvements 83](#_Toc436853977)

[Step 6 – Confirm improvements 83](#_Toc436853978)

[Step 7 – Sustain improvements 83](#_Toc436853979)

[Step 8 – Monitor performance 83](#_Toc436853980)

[Lessons Learned 84](#_Toc436853981)

[Anthony Farina 84](#_Toc436853982)

[Lourdes Lopez 84](#_Toc436853983)

[Joanna Hang 85](#_Toc436853984)

[David Sullivan 85](#_Toc436853985)

[Timothy Cioffi 86](#_Toc436853986)

[Team Experience and Lessons Learned 87](#_Toc436853987)

[Reference list 88](#_Toc436853988)

[Appendix 89](#_Toc436853989)

[Team Charter 90](#_Toc436853990)

[Team Evaluation 94](#_Toc436853991)

# Introduction

Caliware’s software process, composed of Scrum and XP practices, modeled a lightweight software process. In order for our process to comply with ISO 12207, we performed an ISO 15504 compliant assessment of our process against ISO 12207 in order to reach our objective of meeting Capability Level 3 (CL3). To achieve higher capability levels, our organization followed the first four steps of the Process Improvement as followed:

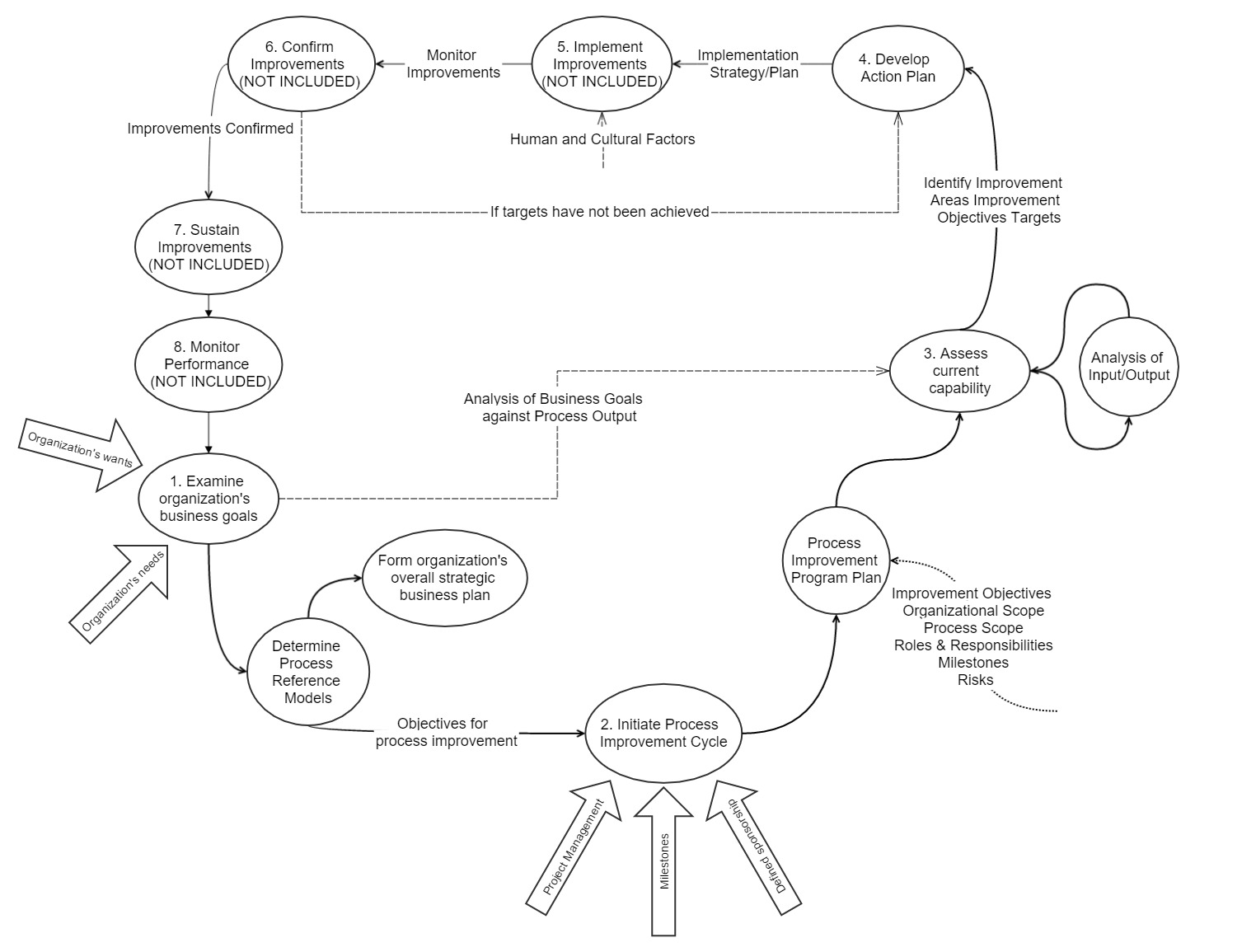


Figure : Steps of Process Improvement

First, we examined our organization’s business goals, which analyzed the organization’s needs. From our goals, we derived the objectives to guide the Process Improvement Cycle.

Second, we initiated the process improvement cycle by producing the Process Improvement Program, which refers to the project management of the program. The Process Improvement Program Plan included the organizational background and history. It also showed the current status of the organizational process improvement activities. It stated the improvement objectives, organizational scope for the program, process scope, process improvement lifecycle, roles and responsibilities, resources, review points, report mechanisms, risks regarding the program, risk management, activities that were needed to be performed of the program, and the progress.

Third, our organization assessed our current capabilities. We compared and mapped the Scrum and XP practices to ISO 12207. Then we performed an ISO 15504 compliant assessment by finding the gaps between our process and ISO 12207 processes. After the gap analysis, we then rated each Process Attribute (PA) and Capability Level (CL) of each process.

Fourth, we developed an action plan. For any processes that could not be rated to CL3, these were our findings needed in order to achieve CL3. These findings were used to build the action plan. Our action plan identified improvement areas, analyzed assessment strengths and weaknesses, reviewed organizational improvement objectives, analyzed effectiveness measurements, listed improvement areas, defined detailed improvement objectives and set targets, and derived an action plan.

Steps five to eight of the Process Improvement Cycle (implement improvements, confirm improvements, sustain improvements, and monitor performance) can be performed at a future date and are not shown in this homework assignment.

# 

# Step 1 – Examine organization‘s business goals

## Business goals

For step one of the process improvement plan, we analyzed our organization’s business goals. Our company follows the principle of achieving high customer satisfaction by offering high quality software products and services. Offering high quality products and services entails our company to be competitive amongst other businesses while maintaining a reputable organization. From these goals, we determined our organization’s needs of establishing better overall development processes, reducing development cost and increasing product quality. For our organization to be successful, our organization needed to reach process CL3 to achieve our client’s capability requirement goal.

## Objectives of Process Improvement

After analyzing the business goals, we initiated the process improvement plan with the objectives listed below. The organization prioritized these objectives based off their importance and impact.

1. Deliver products within budget on time according to schedule
2. Increase overall product quality
3. Improve development efficiency - less rewrite through better requirements analysis
4. Ensure adaptive Life Cycle processes are utilized to be responsive to changing requirements
5. Establish measurement processes with a strategy for continuous process improvement
6. Identify risks to meeting project objectives early
7. Identify process improvement areas to improve in order to deliver higher quality products and increase organizational capability
8. Ensure products adhere to requirements that meet stakeholder’s objectives
9. Monitor progress to goals and avoid risky last minute changes
10. Preserve the integrity of a SW design
11. Improve change management
12. Enable predictive instead of reactive changes to software or systems
13. Build on organization strengths and improve areas of weakness and creation of standards for development processes to increase effectiveness
14. Maintain processes after improvements have been established
15. The plan that is developed can be reused with projects with similar scope and produce consistent results

Overall, these objectives guided the improvement targets for the Process Improvement Plan. These objectives were made aware to the entire organization to gather managerial and financial commitment to undertake a Process Improvement Program. An improvement program budget and improvement priorities were identified.

# Step 2 – Initiate process improvement cycle

The Process Improvement Cycle was implemented as a project known as the Process Improvement Program. In the Process Improvement Program, sponsorship, project management, and budget are defined.

## Process Improvement Program

### Sponsorship

The Process Improvement Cycle Program was initiated by the sponsor, which included all senior management. The senior management assigned a site manager to ensure the success of the assessment and to gather the sufficient resources required for the program (Jo, n.d.-e). The site manager also initiated an open culture within the entire organization into accepting an assessment process (Jo, n.d.-d). The management then delegated a project management team for the Process Improvement Cycle.

### Project Management

With the confidence of the upper management, a project management team was assembled. This team was known as the Software Engineering Process Group (SEPG), and was responsible for the assessment process and process improvement. This group was an informal group, formed internally within the organization. The team leader was a knowledgeable manager, and the assessment team was composed of four software professionals in the organization. These individuals were experienced software developers from various departments and attended assessment training.

### Budget for Process Improvement Cycle

SEGP was composed of five individuals, all of whom had an hourly salary of $40. These individuals completed the training program for the Certification ISO/IEC 15504 and completed the certification exam. Afterwards, SEPG began the Process Improvement Cycle. For each step, the number of hours to be completed were pre-determined. A total number of 288 hours were planned to complete the program with a contingency of an additional 40 hours if needed. The assessment would could $99,225. From this, the budget for the Process Improvement Cycle was calculated and is listed below:

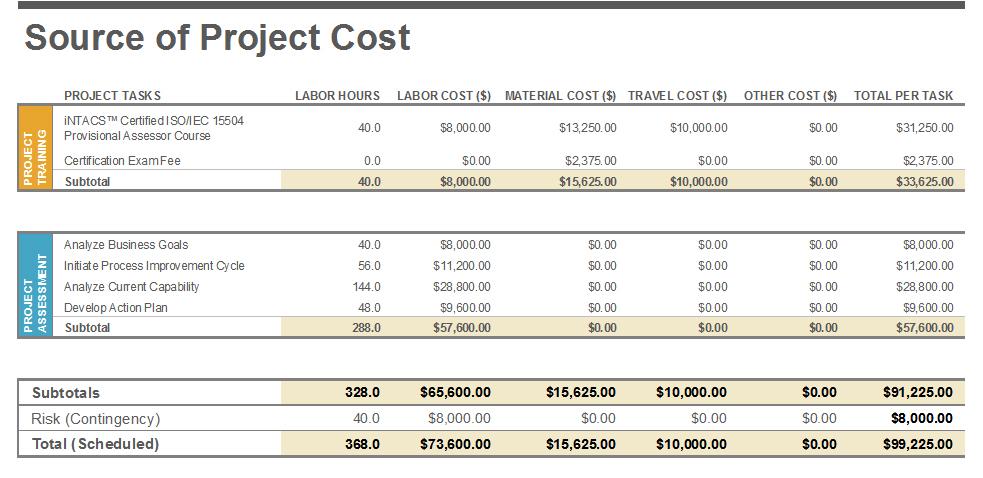


Figure : Cost breakdown

### Accountability

The sponsor was accountable for selecting the Process Reference Model (PRM) and selecting the assessors (SEPG). The sponsor was accountable for ensuring sufficient resources were available to conduct the assessment (Jo, n.d.-c). The sponsor was also accountable for ensuring the assessment team had access to the relevant resources as needed.

SEPG was accountable for its commitment to the assessment and ensured its knowledge about the assessment process and that the assessment be conducted according to the requirements of ISO/IEC 15504. During the assessment, SEPG was accountable for ensuring any participants are briefed about the purpose, scope, and approach of the assessment (Jo, n.d.-e). SEPG was also accountable for delivering the assessment results to the sponsors.

## Process Improvement Program Plan

The Process Improvement Program Plan was produced to monitor the progress of the program and is detailed further below:

### Background, History, Current Status of Organizational Process Improvement Activities

Originally, our organization followed a Scrum and XP lifecycle process. This process was a lightweight and agile model. Our client requested and required that our software process be improved to ISO 12207 compliant process, must maintain Capability Level 2 and target Capability Level 3. An assessment that is compliant with ISO 15504 (SPICE) was required to find the gaps between the current process and ISO 12207 processes.

### Improvement Objectives

The organization understood the need for improvement for a Process Improvement Plan and defined its objective. The improvement objective for the Process Improvement Plan was to implement an ISO 12207 compliant process that can be assessed to meet Capability Level 3 by ISO 15504.

### Organizational Scope

The scope of the program was primarily limited to the software groups including developers, testers, quality assurance. These groups were the technical team of the organization. Therefore, the program excluded other areas of the organization such as human resources, finance department, marketing, and communications.

### Process Scope

The Process Improvement Plan analyzed both system context processes and software specific processes out of ISO 12207, but the scope was limited to the software specific processes for identifying areas of improvement. Specifically, this included software implementation processes and software support processes. All 43 processes were mapped, but only the gaps found in 7.1.2 Software Requirements Analysis Process, 7.1.5 Software Construction Process, 7.2.2 Software Configuration Management Process, and 7.2.3 Software Quality Assurance were analyzed for gaps then rated. Based on any weaknesses or omissions in the current software process, the team provided recommendations of improvements to these processes to improve to Capability Level 3.

### Process Improvement Lifecycle

The Process Improvement Plan covered only steps one through eight of the Process Improvement Cycle listed below:

1. Examine organization’s business goals
2. Initiate Process Improvement Cycle
3. Assess current capability
4. Develop action plan
5. Implement improvements (not included)
6. Confirm improvements (not included)
7. Sustain improvements (not included)
8. Monitor performance (not included)

### Key Roles and Responsibilities

Roles and responsibilities for Process Improvement Program was established.

**Sponsor**

The sponsor supported the Process Improvement Program. The sponsor selected the PRM and appointed competent assessor (Jo, n.d.-c). The sponsor ensured resources available and ensured access to any resources needed for the assessment team.

**Assessment Leader**

The assessment leader was responsible for the assessment team to have the proper knowledge, skills, and resources to perform the assessment activities and to document the activities.

**SEPG**

SEPG consisted of a project manager to lead the process assessment effort and four additional experienced software engineers to perform software assessment activities. Their role was to ensure that the process assessment is properly carried out to the conformance of ISO 15504 and provide status reports. Members acquired the training to implement process assessment.

### Resources

Resources were planned and committed to support the project.

* Comprehensive documentation and training materials
* ISO 12207
* ISO 15504
* Budget for Assessment
* Assessment materials
  + Mapping template
  + Gap template
  + Rating template

### Appropriate Milestones, Review Points, Mechanisms

To ensure the assessment is successful, appropriate milestones and review points were determined. Senior management reviews were bi-weekly. Milestones were set by the each step in the Process Improvement Cycle between dates October 5, 2015 and December 3, 2015. The timeline for the appropriate milestones and a table with detailed dates are shown below:

**Project Timeline**

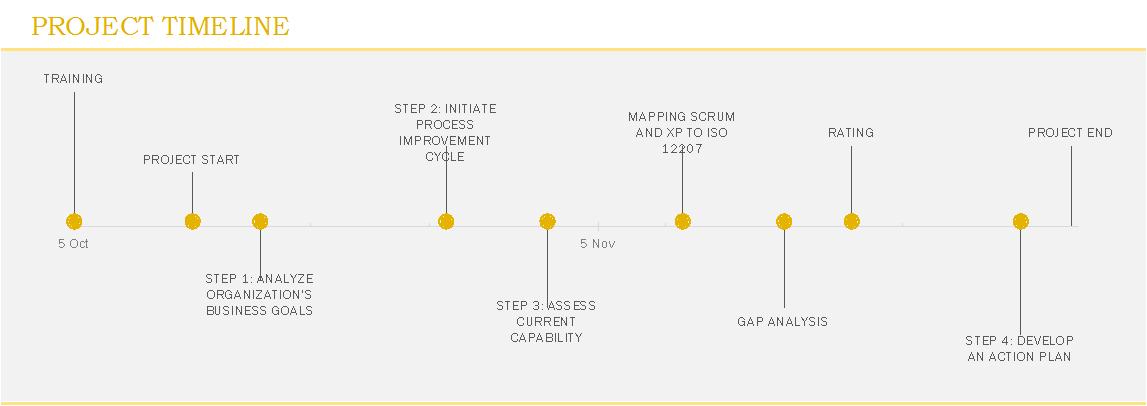


Figure : project Timeline Graph

**Project Timeline Table**

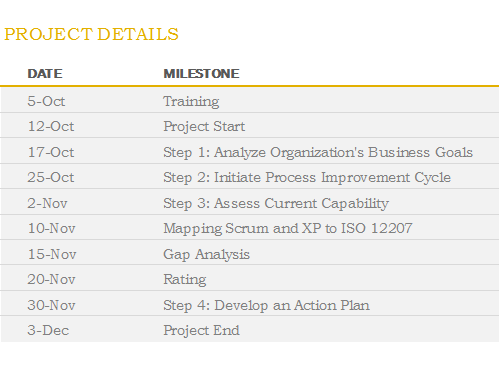


Figure : Project Timeline Table

### Risks Associated with the Program and Risk Management

There were risks associated with the Process Improvement Program and risk management was required to reduce and eliminate these risks. One risk of the Process Improvement Program was the availability of resources. Resources such as staffing and funding were a risk as it could negatively affect the assessment process. To reduce this risk, proper allocation of resources and a budget was required (Jo, n.d.-e). A second risk was having adequate support. Inadequate support would endanger the outcome of the program. To reduce this risk, commitment to the program was made at the highest management level to ensure that the organization understood that the program would be a long-term process (Jo, n.d.-e). The third risk was lack of follow-through or a commitment to see the end of the assessment. Because the program would be a long-term commitment, a site manager who was determined and aggressive to lead the assessment efforts to alleviate this risk (Jo, n.d.-e).

### Assessment Activities Performed

The following were assessment activities that were performed:

1. Performed an assessment based on current practices by comparing and mapping Scrum and XP practices to ISO 12207.
2. Utilized standards and techniques in the 15504 process to determine the capability level to meet at least CL3.
3. Analyzed the weaknesses of the current process and define process improvements by deriving actions from the above standards based on the results of the assessment.
4. Developed an action plan to define the actions to improve the process to meet the target capability model for each process that will be assessed.

### Assessment Outputs

1. Process mapping document
2. Gaps to ISO 12207 document
3. Process assessment rating document
4. Capability Profile
5. Process Capability Level
6. Action plan

# Step 3 – Assess current capability

Step three assessed the organization’s current capability using a 15504 compliant process. This step was completed by mapping Scrum and XP to ISO 12207 for all 43 processes, analyzing the gaps between the processes, and rating the processes.

## Comparison/Mapping (Scrum & XP to ISO 12207)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mapping Template for the Agile Process (Scrum & XP) to ISO 12207** | | | | |
| **Clauses in ISO 12207** | **Processes**  **in ISO 12207** | **Purpose and Outcomes**  **[AND] Activities**  **in ISO 12207** | **Agile Process: Scrum & XP Practices** |  |
|  |  |  |  |  |
| **6.1** | **Agreement Processes** |  |  |  |
| **6.1.1** | **Acquisition Process** | **6.1.1.1 Purpose:**  The purpose of the Acquisition Process is to obtain the product and/or service that satisfies the need expressed by the acquirer. The process begins with the identification of customer needs and ends with the acceptance of the product and/or service needed by the acquirer.  **6.1.1.2 Outcomes:**  As a result of successful implementation of the Acquisition Process:  a) acquisition needs, goals, product and/or service acceptance criteria and acquisition strategies are  defined;  b) an agreement is developed that clearly expresses the expectation, responsibilities and liabilities of both  the acquirer and the supplier;  c) one or more suppliers is selected;  d) a product and/or service is acquired that satisfies the acquirer’s stated need;  e) the acquisition is monitored so that specified constraints such as cost, schedule and quality are met;  f) supplier deliverables are accepted; and  g) Any identified open items have a satisfactory conclusion as agreed to by the acquirer and the supplier. | No match.  Our process does not specify a specific process in the acquisition of products and/or services. However, acquisition may be expressed in the needs of the customer in the User Stories. |  |
| **6.1.1.3.1** |  | **Acquisition preparation** |  |  |
| **6.1.1.3.2** |  | **Acquisition advertisement** | No match |  |
| **6.1.1.3.3** |  | **Supplier selection** | No match |  |
| **6.1.1.3.4** |  | **Contract agreement** | No match |  |
| **6.1.1.3.5** |  | **Agreement monitoring** | No match |  |
| **6.1.1.3.6** |  | **Acquirer acceptance** | No match |  |
| **6.1.1.3.7** |  | **Closure** | No match |  |
| **6.1.2** | **Supply Process** | **6.1.2.1 Purpose**  The purpose of the Supply Process is to provide a product or service to the acquirer that meets the agreed requirements.  **6.1.2.2 Outcomes**  As a result of successful implementation of the Supply Process:  a) an acquirer for a product or service is identified;  b) a response to an acquirer's request is produced;  c) an agreement is established between the acquirer and the supplier for developing, maintaining, operating,  packaging, delivering, and installing the product and/or service;  d) a product and/or service that meets the agreed requirements are developed by the supplier;  e) the product and/or service is delivered to the acquirer in accordance with the agreed requirements; and  f) The product is installed in accordance with the agreed requirements. | No match  Our process does not specify a specific process in supplying of products and/or services. However, supplying products/services may be expressed in the needs of the customer in the User Stories. |  |
| **6.1.2.3.1** |  | **Opportunity identification** | No match |  |
| **6.1.2.3.2** |  | **Supplier tendering** | No match |  |
| **6.1.2.3.3** |  | **Contract agreement** | No match |  |
| **6.1.2.3.4** |  | **Contract execution** | No match |  |
| **6.1.2.3.5** |  | **Product/service delivery and support** | No match |  |
| **6.1.2.3.6** |  | **Closure** | No match |  |
| **6.2** | **Organizational Project-Enabling Processes** |  |  |  |
| **6.2.1** | **Life Cycle Model Management Process** | **6.2.1.1 Purpose**  The purpose of the Life Cycle Model Management Process is to define, maintain, and assure availability of policies, life cycle processes, life cycle models, and procedures for use by the organization with respect to the scope of this International Standard.  This process provides life cycle policies, processes, and procedures that are consistent with the organization's objectives, that are defined, adapted, improved and maintained to support individual project needs within the context of the organization, and that are capable of being applied using effective, proven methods and tools.  **6.2.1.2 Outcomes**  As a result of the successful implementation of the Life Cycle Model Management Process:  a) policies and procedures for the  management and deployment of life cycle models and processes are  provided;  b) responsibility, accountability and authority for life cycle management are defined;  c) life cycle processes, models and procedures for use by the organization are defined, maintained and improved; and  d) Prioritized process improvements are implemented. | We defined our Life Cycle Model by selecting processes from Scrum and XP practices.  As a result of our Live Cycle Model, we successfully implemented:  a) life cycle processes and procedures were defined;  b) Roles and responsibilities were defined.  Responsibility for the outcome was defined. For Scrum, roles were defined as the following Customer/Product Owner, Scrum Team, Scrum Master, and Chicken. In XP, the roles were defined for Customer, Programmer, Tester, Coach, Tracker, and Consultant.  c) process improvements were defined during the Sprint Retrospective meeting to be implemented for the next Sprint;  d) life cycle processes improvements were implemented |  |
| **6.2.1.3.1** |  | **Process establishment** | We defined roles and practices  **Roles:**   * Product Owner/Customer (Scrum & XP) * Scrum Team (Scrum) * Scrum Master (Scrum) * Chicken (Scrum) * Customer (XP) * Programmer(s) (XP) * Tester(s) (XP) * Coach (XP) * Tracker (XP) * Consultant (XP)   **Practices:**   * Scrum   + Pre-Game Planning   + Sprint Planning   + Sprint Review   + High-Level Design Phase   + Sprint   + Sprint Review   + Don’t Add to Iteration   + Chicken and Pigs   + Scrum Master Firewall   + Decisions in 1 Hour   + Blocks Gone in 1 Day   + Teams of 7   + Daily Scrum   + Self-Directed and Self-Organizing Teams   + Daily Build   + Common Room * XP   + Planning Game   + Onsite customer   + Acceptance Testing   + System metaphors   + Frequent Refactoring   + Simple design   + Coding standards   + Pair programming   + Team code ownership   + Customer Tests   + Test-first development unit testing   + Short releases   + Sustainable pace   + Stand-up meeting   + Continuous integration   + Common room |  |
| **6.2.1.3.2** |  | **Process assessment** | At the end of each Sprint Review, a Sprint Retrospective meeting was held. During the meeting, the Scrum Team must review and assess the Sprint by asking the following questions:   * What went well during the sprint? * What went wrong during the sprint? * What could we do differently to improve? |  |
| **6.2.1.3.3** |  | **Process improvement** | The Scrum Team implemented its process assessment by holding the Sprint Retrospective meeting at the end of each Sprint Review and answering the questions listed above. This process was used to identify and implement improvements to the process for the next Sprint. |  |
| **6.2.2** | **Infrastructure Management Process** | **6.2.2.1 Purpose**  The purpose of the Infrastructure Management Process is to provide the enabling infrastructure and services to projects to support organization and project objectives throughout the life cycle.  This process defines, provides and maintains the facilities, tools, and communications and information  technology assets needed for the organization’s business with respect to the scope of this International  Standard.  **6.2.2.2 Outcomes**  As a result of the successful implementation of the Infrastructure Management Process:  a) the requirements for infrastructure to support processes are defined;  b) the infrastructure elements are identified and specified;  c) infrastructure elements are acquired;  d) the infrastructure elements are implemented; and  e) A stable and reliable infrastructure is maintained and improved.  NOTE The infrastructure elements may include hardware, software, methods, tools, techniques, standards, and facilities for development, operation, or maintenance. | Infrastructure facilities include a common room for Scrum and XP.  For XP, techniques and tools were evaluated and selected during the first iteration.  We defined development tools to support our organization and project objectives. For Scrum, daily meetings were held in a physical space shared by all Scrum Team members. For XP, a common room helped customers, developers, and managers deliver software with high business value.  As a result of our infrastructure management, we produced the following:  a) defined our tools to support our processes such as development and testing;  b) identified the necessary tools for development;  c) acquired development tools;  d) implemented the development tools in the first Sprint; |  |
| **6.2.2.3.1** |  | **Process implementation** | We defined the essential tools at the beginning of the first Sprint and during architectural spike. |  |
| **6.2.2.3.2** |  | **Establishment of the infrastructure** | We implemented the tools by installing the required software which was planned for the first sprint. |  |
| **6.2.2.3.3** |  | **Maintenance of the infrastructure** | No match |  |
| **6.2.3** | **Project Portfolio Management Process** | **6.2.3.1 Purpose**  The purpose of the Project Portfolio Management Process is to initiate and sustain necessary, sufficient and suitable projects in order to meet the strategic objectives of the organization.  This process commits the investment of adequate organization funding and resources, and sanctions the authorities needed to establish selected projects. It performs continued qualification of projects to confirm they justify, or can be redirected to justify, continued investment.  **6.2.3.2 Outcomes**  As a result of the successful implementation of the Project Portfolio Management Process:  a) business venture opportunities, investments or necessities are qualified, prioritized and selected;  b) resources and budgets for each project are identified and allocated;  c) project management accountability and authorities are defined;  d) projects meeting agreement and stakeholder requirements are sustained; and  e) Projects not meeting agreement or stakeholder requirements are redirected or terminated. | No match |  |
| **6.2.3.3.1** |  | **Project initiation** | No match |  |
| **6.2.3.3.2** |  | **Portfolio evaluation** | No match |  |
| **6.2.3.3.3** |  | **Project closure** | No match |  |
| **6.2.4** | **Human Resource Management Process** | **6.2.4.1 Purpose**  The purpose of the Human Resource Management Process is to provide the organization with necessary human resources and to maintain their competencies, consistent with business needs.  The process assures the providing of a supply of skilled and experienced personnel qualified to perform life cycle processes to achieve organization, project and customer objectives.  **6.2.4.2 Outcomes**  As a result of the successful implementation of the Human Resource Management Process:  a) skills required by projects are identified;  b) necessary human resources are provided to projects;  c) skills of personnel are developed, maintained or enhanced;  d) conflicts in multi-project resource demands are resolved; and  e) Individual knowledge, information and skills are collected, shared, reused and improved throughout the organization. | The Human Resource Process maps to the Scrum’s and XP’s roles and responsibilities.  a) Scrum did not directly assign tasks. Tasks were volunteered by the individual developers based on their knowledge, experience, and skill.  b) The Scrum Team is a self-organizing team. The combined effort of the Scrum Master and developers shared responsibilities. The Scrum Master supported the team. Developers volunteered.  c) The Scrum Master facilitated the training of the Scrum Team. The Scrum Master ensured the team was following Scrum practices. XP emphasized moving people around to avoid knowledge loss. Cross-training and pair programming helped less knowledgeable developers learn from the experienced developers.  d) Scrum Master removes impediments and acts as a firewall.  e) No match |  |
| **6.2.4.3.1** |  | **Skill identification** | In XP’s Exploration Stage, the team ensures the project is feasible. The team identifies the skills and knowledge needed to complete the project by exploring the proof of technology programming. |  |
| **6.2.4.3.2** |  | **Skill development** | The team acquired training materials by downloading the Android development kit, watching tutorials, learning about Agile processes. Pair programming was done by screen sharing using Google Hangouts |  |
| **6.2.4.3.3** |  | **Skill acquisition and provision** | Scrum has self-directed and self-organizing teams. Thus, the team was empowered in setting their own responsibilities and solving problems. Team members had understanding of their role on the project and shared vision (Scrum), and communicated appropriately. |  |
| **6.2.4.3.4** |  | **Knowledge management** | No match |  |
| **6.2.5** | **Quality Management Process** | **6.2.5.1 Purpose**  The purpose of the Quality Management Process is to assure that products, services and implementations of life cycle processes meet organizational quality objectives and achieve customer satisfaction.  **6.2.5.2 Outcomes**  As a result of the successful implementation of the Quality Management process:  a) organization quality management policies and procedures are defined;  b) organization quality objectives are defined;  c) accountability and authority for quality management are defined;  d) the status of customer satisfaction is monitored; and  e) Appropriate action is taken when quality objectives are not achieved. | During the Sprint Review, a demo of the product was given to the Product Owner/Customer to gather feedback. At the next Sprint Planning meeting, the Product Owner adjusted priorities based on the feedback from what was learned in the previous Sprint. |  |
| **6.2.5.3.1** |  | **Quality management** | No match |  |
| **6.2.5.3.2** |  | **Quality management corrective action** | No match |  |
| **6.3** | **Project Processes** |  |  |  |
| **6.3.1** | **Project Planning** | **6.3.1.1 Purpose**  The purpose of the Project Planning Process is to produce and communicate effective and workable project plans.  This process determines the scope of the project management and technical activities, identifies process outputs, project tasks and deliverables, establishes schedules for project task conduct, including achievement criteria, and required resources to accomplish project tasks.  **6.3.1.2 Outcomes**  As a result of successful implementation of the Project Planning Process:  a) the scope of the work for the project is defined;  b) the feasibility of achieving the goals of the project with available resources and constraints are evaluated;  c) the tasks and resources necessary to complete the work are sized and estimated;  d) interfaces between elements in the project, and with other project and organizational units, are identified;  e) plans for the execution of the project are developed; and  f) Plans for the execution of the project are activated. | In the Pre-Game Planning stage, scope was defined when the Scrum Master and Product owner establish the vision and set expectations for developing the product. The size and estimation were also done by the developers during the Spring Planning meeting where developers break down the requirements and made the decision of what work can be completed in the iteration. Then the tasks were created from the User Stories along with the estimated time for completion of the tasks.  Communication was effective for the Scrum Team because the needs of the stakeholders were addressed through project meetings directed by the Scrum Master.  The creation of the Product Backlog was the basis of planning requirements Scrum Team.  Release Planning established schedules for releases. Deliverables were defined for each release and iteration to be reviewed.  During creation of the prototype or architectural spike, the feasibility of requirements was determined early. |  |
| **6.3.1.3.1** |  | **Project initiation** | Requirements were gathered with the User Stories. In the Pre-Game Planning stage, the team checked feasibility of the project. Any new requirements were accepted. |  |
| **6.3.1.3.2** |  | **Project planning** | The Sprint Planning meeting sets a schedule for the completion of tasks. In the Task Board, the number of hours to complete a task are estimated. The developers volunteer for these tasks and assume the responsibilities for these tasks. |  |
| **6.3.1.3.3** |  | **Project activation** | The Product Owner is defined as the authority for the Product Backlog and obtains approval. The Scrum Master oversees the project’s direction and ensures the Scrum Team is following Scrum procedures. |  |
| **6.3.2** | **Project Assessment and Control Process** | **6.3.2.1 Purpose**  The purpose of the Project Assessment and Control Process is to determine the status of the project and ensure that the project performs according to plans and schedules, and within projected budgets, and that it satisfies technical objectives.  This process includes redirecting the project activities, as appropriate, to correct identified deviations and variations from other project management or technical processes. Redirection may include re-planning as appropriate.  **6.3.2.2 Outcomes**  As a result of the successful implementation of the Project Assessment and Control Process:  a) progress of the project is monitored and reported;  b) interfaces between elements in the project, and with other project and organizational units, are monitored;  c) actions to correct deviations from the plan and to prevent recurrence of problems identified in the project  are taken when project targets are not achieved; and  d) Project objectives are achieved and recorded. | Assessment and control was performed by the Scrum Team by tracking progress of the tasks on the Task Board and the Burndown Chart, reviewing the work during the Sprint Review Meeting, prioritizing the User Stories in agreement with the Product Owner during the Sprint Planning meeting, and reviewing the process during the Sprint Review. |  |
| **6.3.2.3.1** |  | **Project monitoring** | The Task Board was updated during the project to monitor progress by moving the tasks from the To Do, In Progress, and Completed. The Task Board also estimates the number of hours left on a task. From these tasks, the Burndown Chart was updated to measure work remaining against time. The Task Board was shared to the Scrum Team online. |  |
| **6.3.2.3.2** |  | **Project control** | The Scrum Master acts as a firewall to remove any interruptions or blocks. The Scrum Master also makes decisions within one hour. Any blocks that arise in the Scrum Meeting are removed before the next Scrum Meeting.  In addition, any feedback during the end of each Sprint Review meeting is used to help plan for the next sprint. |  |
| **6.3.2.3.3** |  | **Project assessment** | The Scrum Team reviewed the product during the Sprint Review meeting. Since this is done for each iteration, changes were made during the next Sprint Review meeting. The product owner approves the change and ensures that the suggested changes are aligned with the business objectives. |  |
| **6.3.2.3.4** |  | **Project closure** | Closure was determined when all the Product Backlog items are completed at the end of the final release. |  |
| **6.3.3** | **Decision Management Process** | **6.3.3.1 Purpose**  The purpose of the Decision Management Process is to select the most beneficial course of project action where alternatives exist.  This process responds to a request for a decision encountered during the system life cycle, whatever its nature or source, in order to reach specified, desirable or optimized outcomes. Alternative actions are analyzed and a course of action selected and directed. Decisions and their rationale are recorded to support future decision-making.  **6.3.3.2 Outcomes**  As a result of the successful implementation of the Decision Management Process:  a) a decision-making strategy is defined;  b) alternative courses of action are defined;  c) a preferred course of action is selected; and  d) The resolution, decision rationale and assumptions are captured and reported. | No match mostly. The Scrum Master acts as a firewall and makes decisions within one hour. |  |
| **6.3.3.3.1** |  | **Decision planning** | No match |  |
| **6.3.3.3.2** |  | **Decision analysis** | No match |  |
| **6.3.3.3.3** |  | **Decision tracking** | No match |  |
| **6.3.4** | **Risk Management Process** | **6.3.4.1 Purpose**  The purpose of the Risk Management Process is to identify, analyze, treat and monitor the risks continuously.  The Risk Management Process is a continuous process for systematically addressing risk throughout the life cycle of a system or software product or service. It can be applied to risks related to the acquisition, development, maintenance or operation of a system.  **6.3.4.2 Outcomes**  As a result of successful implementation of the Risk Management Process:  a) the scope of risk management to be performed is determined;  b) appropriate risk management strategies are defined and implemented;  c) risks are identified as they develop and during the conduct of the project;  d) risks are analyzed, and the priority in which to apply resources to treatment of these risks is determined;  e) risk measures are defined, applied, and assessed to determine changes in the status of risk and the  progress of the treatment activities; and  f) Appropriate treatment is taken to correct or avoid the impact of risk based on its priority, probability, and consequence or other defined risk threshold. | According the principles of Agile development, which was created in response to evolving and rapidly changing requirements from customers, risks are found early and can be addressed. This process differentiates from the traditional waterfall process, an assumed linear and sequential methodology, which relies too heavily on having a full set of complete and accurate requirements from the beginning of the software development process. Face-to-face communication is deemed more important than completing requirements up front. Since initial requirements are not fully understood, simple requirements in the form of User Stories with frequent communication with stakeholders will lead to a better quality in matching the customers’ needs.  The Scrum Master insured that any risks in the Sprint will be addressed quickly and have the authority to speak for the team when anything critical that could be a barrier to development or should be brought to the manager’s attention.  During the Sprint Planning meeting, goals that are chosen for the iteration are also usually driven by the highest business value and risk. Architectural spike is used to identify areas of high risk. |  |
| **6.3.4.3.1** |  | **Risk management planning** | When feasibility of an item was a concern during the Pre-Game Planning stage, the team allocated time in the first sprint to perform an architectural spike. |  |
| **6.3.4.3.2** |  | **Risk profile management** | No match |  |
| **6.3.4.3.3** |  | **Risk analysis** | Evaluating risks was done during the sprint. This is known as an architectural spike, which is to determine early what the difficult areas are and to reduce the risks of technical problems and verify the reliability of the requirements. |  |
| **6.3.4.3.4** |  | **Risk treatment** | When a risk was identified, during the daily sprint meeting, the issue was brought to attention and discussed alternative solutions or the possibility of making required design changes. |  |
| **6.3.4.3.5** |  | **Risk monitoring** | No match |  |
| **6.3.4.3.6** |  | **Risk management process evaluation** | During the Sprint Retrospective meeting, the team evaluates the sprint and asks if any improvements are to be made. |  |
| **6.3.5** | **Configuration Management Process** | **6.3.5.1 Purpose**  The purpose of the Configuration Management Process is to establish and maintain the integrity of all identified outputs of a project or process and make them available to concerned parties.  **6.3.5.2 Outcomes**  As a result of the successful implementation of the Configuration Management Process:  a) a configuration management strategy is defined;  b) items requiring configuration management are defined;  c) configuration baselines are established;  d) changes to items under configuration management are controlled;  e) the configuration of released items is controlled; and  f) The status of items under configuration management is made available throughout the life cycle. | No match |  |
| **6.3.5.3.1** |  | **Configuration management planning** | No match |  |
| **6.3.5.3.2** |  | **Configuration management execution** | No match |  |
| **6.3.6** | **Information Management Process** | **6.3.6.1 Purpose**  The purpose of the Information Management Process is to provide relevant, timely, complete, valid and, if required, confidential information to designated parties during and, as appropriate, after the system life cycle.  This process generates, collects, transforms, retains, retrieves, disseminates and disposes of information. It manages designated information, including technical, project, organizational, agreement and user information.  **6.3.6.2 Outcomes**  As a result of the successful implementation of the Information Management Process:  a) information to be managed is identified;  b) the forms of the information representations are defined;  c) information is transformed and disposed of as required;  d) the status of information is recorded;  e) information is current, complete and valid; and  f) Information is made available to designated parties.  NOTE The Software Documentation Management Process is a specialization of the Information Management  Process and is included in the Software Support Process Group. | No match |  |
| **6.3.6.3.1** |  | **Information management planning** | No match |  |
| **6.3.6.3.2** |  | **Information management execution** | No match |  |
| **6.3.7** | **Measurement Process** | **6.3.7.1 Purpose**  The purpose of the Measurement Process is to collect, analyze, and report data relating to the products developed and processes implemented within the organizational unit, to support effective management of the processes, and to objectively demonstrate the quality of the products.  **6.3.7.2 Outcomes**  As a result of successful implementation of the Measurement Process:  a) the information needs of technical and management processes are identified;  b) an appropriate set of measures, driven by the information needs are identified and/or developed;  c) measurement activities are identified and planned;  d) the required data are collected, stored, analyzed, and the results interpreted;  e) information products are used to support decisions and provide an objective basis for communication;  f) the Measurement Process and measures are evaluated; and  g) Improvements are communicated to the Measurement Process owner. | No match |  |
| **6.3.7.3.1** |  | **Measurement planning** | No match |  |
| **6.3.7.3.2** |  | **Measurement performance** | No match |  |
| **6.3.7.3.3** |  | **Measurement evaluation** | No match |  |
| **6.4** | **Technical Processes** |  |  |  |
| **6.4.1** | **Stakeholder Requirements Definition Process** | **6.4.1.1 Purpose**  The purpose of the Stakeholder Requirements Definition Process is to define the requirements for a system that can provide the services needed by users and other stakeholders in a defined environment.  It identifies stakeholders, or stakeholder classes, involved with the system throughout its life cycle, and their needs and desires. It analyzes and transforms these into a common set of stakeholder requirements that express the intended interaction the system will have with its operational environment and that are the reference against which each resulting operational service is validated in order to confirm that the system fulfils needs.  **6.4.1.2 Outcomes**  As a result of successful implementation of the Stakeholder Requirements Definition Process:  a) the required characteristics and context of use of services are specified;  b) the constraints on a system solution are defined;  c) traceability of stakeholder requirements to stakeholders and their needs is achieved;  d) the basis for defining the system requirements is described;  e) the basis for validating the conformance of the services is defined; and  f) A basis for negotiating and agreeing to supply a service or product is provided. | Roles were defined. Stakeholder requirements were managed by the Product Owner through the Product Backlog that contains the User Stories written by the Users. The Product Owner set the priorities and made changes to the Product Backlog as the items evolve during the iterative development process. The User Stories contain the person who wrote the User Story and Scrum Master insures that this contact is available for clarification for the development team. |  |
| **6.4.1.3.1** |  | **Stakeholder identification** | The stakeholders were identified by establishing roles which included:   * Scrum Master (Scrum) * Development Team (Scrum & XP) * Programmer(s) (XP) * Tester(s) (XP) * Product Owner/Customer (Scrum & XP)   In addition, for each User Story, the author of the User Story can be identified. |  |
| **6.4.1.3.2** |  | **Requirements identification** | Requirements are added via User Stories which are placed in the Product Backlog. Constraints can also be added in the User Story. |  |
| **6.4.1.3.3** |  | **Requirements evaluation** | Missing requirements were identified analyzed during the Sprint. During the Spring Planning meeting, requirements are prioritized and ranked. |  |
| **6.4.1.3.4** |  | **Requirements agreement** | User Stories were decided during the Sprint Planning meeting. The Scrum Team discussed what will be included and negotiated what can be done to meet the Sprint Goal with the Product Owner. |  |
| **6.4.1.3.5** |  | **Requirement recording** | Requirements were recorded in the form of User Stories and then added to the Product Backlog. These are traceable to the author that wrote the User Story. |  |
| **6.4.2** | **System Requirements Analysis Process** | **6.4.2.1 Purpose**  The purpose of System Requirements Analysis is to transform the defined stakeholder requirements into a set of desired system technical requirements that will guide the design of the system.  **6.4.2.2 Outcomes**  As a result of successful implementation of System Requirements Analysis:  a) a defined set of system functional and non-functional requirements describing the problem to be solved  are established;  b) the appropriate techniques are performed to optimize the preferred project solution;  c) system requirements are analyzed for correctness and testability;  d) the impact of the system requirements on the operating environment are understood;  e) the requirements are prioritized, approved and updated as needed;  f) consistency and traceability are established between the system requirements and the customer’s  requirements baseline;  g) changes to the baseline are evaluated for cost, schedule and technical impact; and  h) The system requirements are communicated to all affected parties and baselined. | No match |  |
| **6.4.2.3.1** |  | **Requirements specification** |  |  |
| **6.4.2.3.2** |  | **Requirements evaluation** |  |  |
| **6.4.3** | **System Architectural Design Process** | **6.4.3.1 Purpose**  The purpose of the System Architectural Design Process is to identify which system requirements should be allocated to which elements of the system.  **6.4.3.2 Outcomes**  As a result of successful implementation of the System Architectural Design Process:  a) a system architecture design is defined that identifies the elements of the system and meets the defined  requirements;  b) the system’s functional and non-functional requirements are addressed;  c) the requirements are allocated to the elements of the system;  d) internal and external interfaces of each system element are defined;  e) verification between the system requirements and the system architecture is performed;  f) the requirements allocated to the system elements and their interfaces are traceable to the customer’s  requirements baseline;  g) consistency and traceability between the system requirements and system architecture design is  maintained; and  h) the system requirements, the system architecture design, and their relationships are baselined and  communicated to all affected parties;  i) human factors and ergonomic knowledge and techniques are incorporated in system design; and  j) human-centered design activities are identified and performed. | No match |  |
| **6.4.3.3.1** |  | **Establishing architecture** |  |  |
| **6.4.3.3.2** |  | **Architectural evaluation** |  |  |
| **6.4.4** | **Implementation Process** | **6.4.4.1 Purpose**  The purpose of the Implementation Process is to realize a specified system element.  (No specified outcomes in document) | No match |  |
| **6.4.5** | **System Integration Process** | **6.4.5.1 Purpose**  The purpose of the System Integration Process is to integrate the system elements (including software items, hardware items, manual operations, and other systems, as necessary) to produce a complete system that will satisfy the system design and the customers’ expectations expressed in the system requirements.  **6.4.5.2 Outcomes**  As a result of successful implementation of the System Integration Process:  a) a strategy is developed to integrate the system according to the priorities of the system requirements;  b) criteria are developed to verify compliance with the system requirements allocated to the system  elements, including the interfaces between system elements;  c) the system integration is verified using the defined criteria;  d) a regression strategy is developed and applied for re-testing the system when changes are made;  e) consistency and traceability are established between the system design and the integrated system  elements;  f) an integrated system is constructed that demonstrates compliance with the system design; and  g) An integrated system is constructed that demonstrates that a complete set of usable deliverable system elements exists. | No match |  |
| **6.4.5.3.1** |  | **Integration** |  |  |
| **6.4.5.3.2** |  | **Test readiness** |  |  |
| **6.4.6** | **System Qualification Testing Process** | **6.4.6.1 Purpose**  The purpose of the Systems Qualification Testing Process is to ensure that the implementation of each system requirement is tested for compliance and that the system is ready for delivery.  **6.4.6.2 Outcomes**  As a result of successful implementation of Systems Qualification Testing Process:  a) criteria for evaluating compliance with system requirements are developed;  b) the integrated system is tested using the defined criteria;  c) test results are recorded; and  d) Readiness of the system for delivery is assured. | The test engineer created unit tests for all functions.  The work of the Sprint was verified by the scrum team at the sprint review.  New code was integrated with the project using GitHub and tested together continuously. |  |
| **6.4.6.3.1** |  | **Qualification testing** | Unit tests were created |  |
| **6.4.7** | **Software Installation Process** | **6.4.7.1 Purpose**  The purpose of the Software Installation Process is to install the software product that meets the agreed requirements in the target environment.  **6.4.7.2 Outcomes**  As a result of successful implementation of the Software Installation Process:  a) a software installation strategy is developed;  b) criteria for software installation are developed that demonstrate compliance with the software installation requirements;  c) the software product is installed in the target environment; and  d) Readiness of the software product for use in its intended environment is assured. | No match |  |
| **6.4.7.3.1** |  | **Software installation** |  |  |
| **6.4.8** | **Software Acceptance Support Process** | **6.4.8.1 Purpose**  The purpose of the Software Acceptance Support Process is to assist the acquirer to achieve confidence that the product meets requirements.  **6.4.8.2 Outcomes**  As a result of the successful implementation of the Software Acceptance Support Process:  a) the product is completed and delivered to the acquirer;  b) acquirer acceptance tests and reviews are supported;  c) the product is put into operation in the customers’ environment; and  d) Problems detected during acceptance are identified and communicated to those responsible for resolution.  NOTE Incremental delivery would be in completed increments. | No match |  |
| **6.4.8.3.1** |  | **Software acceptance support.** |  |  |
| **6.4.9** | **Software Operation Process** | **6.4.9.1 Purpose**  The purpose of the Software Operation Process is to operate the software product in its intended environment and to provide support to the customers of the software product.  **6.4.9.2 Outcomes**  As a result of the successful implementation of the Software Operation Process:  a) an operation strategy is defined;  b) conditions for correct operation of the software in its intended environment are identified and evaluated;  c) the software is tested and determined to operate in its intended environment;  d) the software is operated in its intended environment; and  e) Assistance and consultation is provided to the customers of the software product in accordance with the agreement. | No match |  |
| **6.4.9.3.1** |  | **Preparation for operation** |  |  |
| **6.4.9.3.2** |  | **Operation activation and check-out** |  |  |
| **6.4.9.3.3** |  | **Operational use** |  |  |
| **6.4.9.3.4** |  | **Customer support** |  |  |
| **6.4.9.3.5** |  | **Operation problem resolution** |  |  |
| **6.4.10** | **Software Maintenance Process** | **6.4.10.1 Purpose**  The purpose of the Software Maintenance Process is to provide cost-effective support to a delivered software product.  NOTE Pre-delivery Software Maintenance activities include planning for post-delivery operations, supportability, and logistics determination. Post-delivery activities include software modification and operational support, such as training or operating a help desk.  **6.4.10.2 Outcomes**  As a result of successful implementation of the Software Maintenance Process:  a) a maintenance strategy is developed to manage modification and migration of products according to the release strategy;  b) the impact of changes to the existing system on organization, operations or interfaces are identified;  c) affected system and software documentation is updated as needed;  d) modified products are developed with associated tests that demonstrate that requirements are not compromised;  e) product upgrades are migrated to the customer’s environment; and  f) The system software modification is communicated to all affected parties. | No match |  |
| **6.4.10.3.1** |  | **Process implementation** | No match |  |
| **6.4.10.3.2** |  | **Problem and modification analysis** | No match |  |
| **6.4.10.3.3** |  | **Modification implementation** | No match |  |
| **6.4.10.3.4** |  | **Maintenance review/acceptance** | No match |  |
| **6.4.10.3.5** |  | **Migration** | No match |  |
| **6.4.11** | **Software Disposal Process** | **6.4.11.1 Purpose**  The purpose of the Software Disposal Process is to end the existence of a system’s software entity.  This process ends active support by the operation and maintenance organization, or deactivates, disassembles and removes the affected software products, consigning them to a final condition and leaving the environment in an acceptable condition. This process destroys or stores system software elements and related products in a sound manner, in accordance with legislation, agreements, organizational constraints and stakeholder requirements. Where required, it maintains records that may be monitored.  NOTE The objective is to retire a system's existing software products or services while preserving the integrity of organizational operations.  **6.4.11.2 Outcomes**  As a result of successful implementation of the Software Disposal Process:  a) a software disposal strategy is defined;  b) disposal constraints are provided as inputs to requirements;  c) the system's software elements are destroyed or stored;  d) the environment is left in an agreed-upon state; and  e) Records allowing knowledge retention of disposal actions and any analysis of long-term impacts are available. | No match |  |
| **6.4.11.3.1** |  | **Software disposal planning** | No match |  |
| **6.4.11.3.2** |  | **Software disposal execution** | No match |  |
| **7** | **Software Specific Processes** |  |  |  |
| **7.1** | **Software Implementation Processes** |  |  |  |
| **7.1.1** | **Software Implementation Process** | **7.1.1.1 Purpose**  The purpose of the Software Implementation Process is to produce a specified system element implemented as a software product or service.  This process transforms specified behavior, interfaces and implementation constraints into actions that create a system element implemented as a software product or service, otherwise known as a "software item."  This process results in a software item that satisfies architectural design requirements through verification and stakeholder requirements through validation.  **7.1.1.2 Outcomes**  As a result of the successful implementation of the Software Implementation Process:  a) an implementation strategy is defined;  b) implementation technology constraints on the design are identified;  c) a software item is realized; and  d) A software item is packaged and stored in accordance with an agreement for its supply. | Prior to beginning a project, the software implementation process was defined using the Scrum and XP development lifecycle process.  The developers learned about the Scrum and XP process from the Scrum Master and the developers read about the Scrum and XP concepts standards.  **7.1.1.2 Outcomes**  a) A combination of XP and Scrum concepts were used to define the implementation strategy.  b) During the planning phase potential technology constraints are identified.  c) Software items were constructed during the development iteratively through sprints.  d) Software items were delivered in accordance to the release plan. |  |
| **7.1.1.3.1** |  | **Software implementation strategy** | Many Scrum and XP Concepts were used to document the software development process such as the product backlog, user stories, tasks, sprint burndown chart and release plan but these concepts do not map to a fully documented implementation strategy. |  |
| **7.1.2** | **Software Requirements Analysis Process** | **7.1.2.1 Purpose**  The purpose of Software Requirements Analysis Process is to establish the requirements of the software elements of the system.  **7.1.2.2 Outcomes**  As a result of successful implementation of the Software Requirements Analysis Process:  a) the requirements allocated to the software elements of the system and their interfaces are defined;  b) software requirements are analyzed for correctness and testability;  c) the impact of software requirements on the operating environment are understood;  d) consistency and traceability are established between the software requirements and system  requirements;  e) prioritization for implementing the software requirements is defined;  f) the software requirements are approved and updated as needed;  g) changes to the software requirements are evaluated for cost, schedule and technical impact; and  h) The software requirements are baselined and communicated to all affected parties. | The software requirements Analysis Process 7.1.2 mapped to Scrum pre-game planning, sprint planning, and sprint review, as well as XP’s story cards, acceptance testing and onsite customer.  **Outcomes:**  a) Requirements were gathered and documented into stories and organized into a Product Backlog.  b) The Product Backlog was developed in collaboration with the product owner for accuracy in the requirements. Stories contain acceptance criteria for testability. Requirements were also reviewed during the sprint review.  c) While the developers estimated the time required to complete a story, and during the creation of the tasks, the requirements impact to the environment was discussed.  d) No match  e) The Product Backlog was prioritized and was refined through each iteration as needed during the sprint planning.  f) After each iteration the product backlog was reviewed with the product owner and updated if needed.  g) Analysis for the amount of work was done by asking the developers to break down the user stories into tasks with estimates of each task. Changes to the requirements were evaluated for cost, schedule and technical impact as the developers estimated time for the stories.  h) The product backlog is the central requirements document and is openly accessible to the team. User stories were added to the Product Backlog before the work is done in the Sprint iterations. |  |
| **7.1.2.3.1** |  | **Software requirements analysis** | : **7.1.2.3.1.1**  Activities from Scrum and XP such as creating user stories, developing and maintaining a product backlog, developing prototypes and exploratory design and ensuring fusibility were conducted in order to gather and analyze software requirements. User stories captured functional requirements and the acceptance criteria captured various other characteristics such as performance requirements and security requirements. The scrum team refined the user stories and discussed what needed to be developed. User stories were maintained in the Product Backlog and were owned by the Product Owner. The user stories were prioritized updated during development. |  |
| **7.1.3** | **Software Architectural Design Process** | **7.1.3.1 Purpose**  The purpose of the Software Architectural Design Process is to provide a design for the software that implements and can be verified against the requirements.  **7.1.3.2 Outcomes**  As a result of successful implementation of the Software Architectural Design Process:  a) a software architectural design is developed and baselined that describes the software items that will  implement the software requirements;  b) internal and external interfaces of each software item are defined; and  c) Consistency and traceability are established between software requirements and software design. | The software Architectural design process partially maps to XP architectural spike concept. Potential architectural spikes were accounted for during the first iteration of the project in order to address architectural design risks, however an architectural design is not fully documented or base lined.  a)No match  b) Internal and external interfaces are identified during the planning phase and are defined iteratively throughout the development phase. Potential architectural risks may be addressed during the architectural spike.  c) Traceability for architectural design was not established. |  |
| **7.1.3.3.1** |  | **Software architectural design** | Partial match. Architectural design is addressed but not fully documented. |  |
| **7.1.4** | **Software Detailed Design Process** | **7.1.4.1 Purpose**  The purpose of the Software Detailed Design Process is to provide a design for the software that implements and can be verified against the requirements and the software architecture and is sufficiently detailed to permit coding and testing.  **7.1.4.2 Outcomes**  As a result of successful implementation of the Software Detailed Design Process:  a) a detailed design of each software component, describing the software units to be built, is developed;  b) external interfaces of each software unit are defined; and  c) Consistency and traceability are established between the detailed design and the requirements and architectural design. | The team’s process does not involve very detailed design. The only preliminary design that is practiced is very basic sketches or prototypes and even verbal conversations.  User stories are created and broken down into tasks. These user stories will satisfy the requirements. No design decisions are included in these user stories. The design is up to the developer but they must adhere to the agreed upon conventions.  Little to any documentation  Developers start work as quickly as possible with little to no design. |  |
| **7.1.4.3.1** |  | **Software detailed design** | :  Sketches, prototypes, verbal conversations. Very basic detail. |  |
| **7.1.5** | **Software Construction Process** | **7.1.5.1 Purpose**  The purpose of the Software Construction Process is to produce executable software units that properly reflect the software design.  **7.1.5.2 Outcomes**  As a result of successful implementation of Software Construction Process:  a) verification criteria are defined for all software units against their requirements;  b) software units defined by the design are produced;  c) consistency and traceability are established between software units and requirements and design; and  d) Verification of the software units against the requirements and the design is accomplished. | **Development Phase Iteration/Sprint**  The software construction process maps to the development phase in both scrum and XP. Development was performed within iterations.  a) Acceptance criteria were documented in the user stories.  b) Software was developed per story and was designed to meet the stories acceptance criteria.  c) Software units were partially traceable to user requirements through the Product Backlog. Coding standards in XP provided consistency and allowed for easier refactoring.  d) Unit tests were performed in order to verify acceptance criteria. |  |
| **7.1.5.3.1** |  | **Software construction** | :  7.1.5.3.1.1  Software and unit tests were developed.  7.1.5.3.1.2  Unit tests were performed in order to ensure that requirements were met.  7.1.5.3.1.3  No match  7.1.5.3.1.4  XP - Continuous integration was performed after unit tests were successful.  7.1.5.3.1.5  Evaluation of code was conducted at the time of development as the code was written by a pair of developers. Unit tests were developed as the code was developed. User stories partially provide traceability from the software item to the requirements and design. Pair programming and code standards were used to ensure consistency within the code construction. A constant review of the user stories and product backlog by the scrum team was used to ensure consistency within the user requirements. XP continuous integration was used to review feasibility of software integration. |  |
| **7.1.6** | **Software Integration Process** | **7.1.6.1 Purpose**  The purpose of the Software Integration Process is to combine the software units and software components, producing integrated software items, consistent with the software design, that demonstrate that the functional and non-functional software requirements are satisfied on an equivalent or complete operational platform.  **7.1.6.2 Outcomes**  As a result of successful implementation of the Software Integration Process:  a) an integration strategy is developed for software units consistent with the software design and the prioritized software requirements;  b) verification criteria for software items are developed that ensure compliance with the software  requirements allocated to the items;  c) software items are verified using the defined criteria;  d) software items defined by the integration strategy are produced;  e) results of integration testing are recorded;  f) consistency and traceability are established between software design and software items; and  g) A regression strategy is developed and applied for re-verifying software items when a change in software units (including associated requirements, design and code) occur. | **Acceptance Testing**  Functional and non-functional testing is also used in XP for acceptance testing within iterative development. Testing is used to verify the software follows the requirements set by the user. The tests are engineered under a certain base requirements tests and are branched off into higher levels of testing, eventually leading to systems testing and over all software testing.  With each tests, a requirement of that iteration is verified by testing with a created user’s inputs and use of the program. This ensures that the program is created towards the standards of the customer.  Testing also allows for a smooth transition between iterations as they will point out requirements faults of the system that need to be fixed for the next iteration, or give verification to the development work that the requirements have been full filled.  Testing outcomes are recorded for use if future bugs arise. |  |
| **7.1.6.3.1** |  | **Software integration** |  |  |
| **7.1.7** | **Software Qualification Testing Process** | **7.1.7.1 Purpose**  The purpose of the Software Qualification Testing Process is to confirm that the integrated software product meets its defined requirements.  **7.1.7.2 Outcomes**  As a result of successful implementation of the Software Qualification Testing Process:  a) criteria for the integrated software is developed that demonstrates compliance with the software  requirements;  b) integrated software is verified using the defined criteria;  c) test results are recorded; and  d) A regression strategy is developed and applied for re-testing the integrated software when a change in software items is made. | **Acceptance Testing**  Software Qualification Testing Process maps to the acceptance testing part of XP. After an iteration, the latest version of the software is tested to verify the quality of the newest version of the software with unit tests. If the newest version fails the test it is sent back to be fixed in the next iteration, if it passes then development continues in the next iteration.  **Outcomes**  Unit tests allow for the user’s qualifications of the software to be proven to work. |  |
| **7.1.7.3.1** |  | **Software qualification testing** | XP - Software unit tests are written and then the code is developed. The other developers understand the code by reviewing the tests. |  |
| **7.2** | **Software Support Processes** |  |  |  |
| **7.2.1** | **Software Documentation Management Process** | **7.2.1.1 Purpose**  The purpose of the Software Documentation Management Process is to develop and maintain the recorded software information produced by a process.  **7.2.1.2 Outcomes**  As a result of successful implementation of the Software Documentation Management Process:  a) a strategy identifying the documentation to be produced during the life cycle of the software product or  service is developed;  b) the standards to be applied for the development of the software documentation are identified;  c) documentation to be produced by the process or project is identified;  d) the content and purpose of all documentation is specified, reviewed and approved;  e) documentation is developed and made available in accordance with identified standards; and  f) Documentation is maintained in accordance with defined criteria. | The software documentation management process is partially addressed by the various work products that are produced while following scrum and XP methodologies, however a formal strategy for maintaining and storing software documentation such as user manuals does not map to Scrum and XP concepts.  **Outcomes**  a) Scrum and XP documentation to be produced throughout the development process were identified prior to the start of the project. Product backlog, user stories and a sprint burn down chart are examples of documentation.  b) User stories are written in accordance with the standard format.  c) Product backlog, user stories and tasks are examples of documentation produced during the development of the software product.  d) Product backlog, user stories and sprint backlog were frequently reviewed before each iteration or during the daily sprint meeting depending on document type.  e)No match  f)No match |  |
| **7.2.1.3.1** |  | **Process implementation** | No match |  |
| **7.2.1.3.2** |  | **Design and development** | No match |  |
| **7.2.1.3.3** |  | **Production** | No match |  |
| **7.2.1.3.4** |  | **Maintenance** | No match |  |
| **7.2.2** | **Software Configuration Management Process** | **7.2.2.1 Purpose**  The purpose of the Software Configuration Management Process is to establish and maintain the integrity of the software items of a process or project and make them available to concerned parties.  **7.2.2.2 Outcomes**  As a result of successful implementation of the Software Configuration Management Process:  a) a software configuration management strategy is developed;  b) items generated by the process or project are identified, defined and baselined;  c) modifications and releases of the items are controlled;  d) modifications and releases are made available to affected parties;  e) the status of the items and modifications are recorded and reported;  f) the completeness and consistency of the items is ensured; and  g) The storage, handling and delivery of the items are controlled. | Scrum and XP process were mapped to software configuration management through the following processes: continuous integration, pair programming, and collective ownership, release planning.  a) Partially mapped to continuous integration concept from XP. Changes were controlled by only allowing one pair to check in at a time. There was a dedicated computer that stored the source code.  b) No match  c) Releases were completed at end of each iteration. Change requests for modifications are reviewed during the sprint planning.  d) The releases were delivered to the stakeholders as scheduled within the release plan.  e) The status of the stories in the iteration was tracked using a burndown chart. The status of the release was given during the sprint review at the end of each iteration.  f) Stories were compared against acceptance criteria to ensure completeness. Consistency was ensured by developing the code using pair programming and by following the standards.  g) No match |  |
| **7.2.2.3.1** |  | **Process implementation** | No match |  |
| **7.2.2.3.2** |  | **Configuration identification** | No match |  |
| **7.2.2.3.3** |  | **Configuration control** | No match |  |
| **7.2.2.3.4** |  | **Configuration status accounting** | The status was tracked using the burndown chart and XP product velocity and were reported to the stakeholders during the sprint review. |  |
| **7.2.2.3.5** |  | **Configuration evaluation** | Developers checked for completeness of the requirements upon completing the stories. Acceptance testing was conducted to ensure that the software meets the requirements. |  |
| **7.2.2.3.6** |  | **Release management and delivery** | The code is maintained for the releases as a master copy. |  |
| **7.2.3** | **Software Quality Assurance Process** | **7.2.3.1 Purpose**  The purpose of the Software Quality Assurance Process is to provide assurance that work products and processes comply with predefined provisions and plans.  **7.2.3.2 Outcomes**  As a result of successful implementation of the Software Quality Assurance Process:  a) a strategy for conducting quality assurance is developed;  b) evidence of software quality assurance is produced and maintained;  c) problems and/or non-conformance with requirements are identified and recorded; and  d) Adherence of products, processes and activities to the applicable standards, procedures and requirements are verified. | Pair programming, collective code ownership and acceptance testing concepts from XP partially map to quality assurance but do not fully encompass a quality assurance process.  Pair programming and collective code ownership help ensure that code is developed to meet the team standards. Acceptance testing ensures that software is written to meet the requirements. Bug items are documented if a requirement is not met correctly.  a) Partially maps to Scrum retrospective meetings which were conducted after each iteration. The iteration was reviewed to identify possible improvements in the process.  b) No match  c) Partially mapped to Scrum retrospective. Issues in the process were identified and recorded.  d) Partially maps to Scrum Master as the Scrum Master verified that the process is being followed correctly. Scrum Master should be trained. |  |
| **7.2.3.3.1** |  | **Process implementation** | No match |  |
| **7.2.3.3.2** |  | **Product assurance** | Pair programming, collective code ownership, refactoring, continuous integration and test driven development partially map and were used to ensure the quality of the product. |  |
| **7.2.3.3.3** |  | **Process assurance** | Scrum Master verifies that the process is being followed during the software development life cycle. |  |
| **7.2.3.3.4** |  | **Assurance of quality systems** | No match |  |
| **7.2.4** | **Software Verification Process** | **7.2.4.1 Purpose**  The purpose of the Software Verification Process is to confirm that each software work product and/or service of a process or project properly reflects the specified requirements.  **7.2.4.2 Outcomes**  As a result of successful implementation of the Software Verification Process:  a) a verification strategy is developed and implemented;  b) criteria for verification of all required software work products is identified;  c) required verification activities are performed;  d) defects are identified and recorded; and  e) Results of the verification activities are made available to the customer and other involved parties. | **Unit Tests**  A practice for software verification that is done in the hybrid scrum/xp process is unit testing. Unit testing allows the team to make sure the product is performing like it is supposed to perform. |  |
| **7.2.4.3.1** |  | **Process implementation** | Unit tests are created first before coding begins |  |
| **7.2.4.3.2** |  | **Verification** | If the unit tests are completed without any errors, the product/code/requirement is verified. |  |
| **7.2.5** | **Software Validation Process** | **7.2.5.1 Purpose**  The purpose of the Software Validation Process is to confirm that the requirements for a specific intended use of the software work product are fulfilled.  **7.2.5.2 Outcomes**  As a result of successful implementation of the Software Validation Process:  a) a validation strategy is developed and implemented;  b) criteria for validation of all required work products are identified;  c) required validation activities are performed;  d) problems are identified and recorded;  e) evidence is provided that the software work products as developed are suitable for their intended use;  and  f) Results of the validation activities are made available to the customer and other involved parties. | **User Story Acceptance Criteria**  Acceptance tests are the main source of software validation. These tests are created from the user stories. These tests are run after every iteration. A user story is not complete/validated until its acceptance test(s) passes. |  |
| **7.2.5.3.1** |  | **Process implementation** | Acceptance tests created in the planning stage with each requirement. They are completed after every iteration. |  |
| **7.2.5.3.2** |  | **Validation** | The requirements or software product is validated after all acceptance tests have passed. |  |
| **7.2.6** | **Software Review Process** | **7.2.6.1 Purpose**  The purpose of the Software Review Process is to maintain a common understanding with the stakeholders of the progress against the objectives of the agreement and what should be done to help ensure development of a product that satisfies the stakeholders. Software reviews are at both project management and technical levels and are held throughout the life of the project.  **7.2.6.2 Outcomes**  As a result of successful implementation of the Software Review Process:  a) management and technical reviews are held based on the needs of the project;  b) the status and products of an activity of a process are evaluated through review activities;  c) review results are made known to all affected parties;  d) action items resulting from reviews are tracked to closure; and  e) Risks and problems are identified and recorded. | **Sprint Review**  In our Scrum/XP process we practiced sprint reviews. During the sprint review, the product is reviewed. The team, product owner, and stakeholders are all present and provide feedback and suggestions. All involved parties would be told of the status of the project along with any problems that have arisen. These reviews are done after all iterations.  Spring meetings also had some form of review which included making problems known and putting them in top priority.  Pair programming was also a part of software review. |  |
| **7.2.6.3.1** |  | **Process implementation** | No match |  |
| **7.2.6.3.2** |  | **Project Management Reviews** | No match |  |
| **7.2.6.3.3** |  | **Technical Reviews** | No match |  |
| **7.2.7** | **Software Audit Process** | **7.2.7.1 Purpose**  The purpose of the Software Audit Process is to independently determine compliance of selected products and processes with the requirements, plans and agreement, as appropriate.  **7.2.7.2 Outcomes**  As a result of successful implementation of the Software Audit Process:  a) an audit strategy is developed and implemented;  b) compliance of selected software work products and/or services or processes with requirements, plans  and agreement is determined according to the audit strategy;  c) audits are conducted by an appropriate independent party; and  d) Problems detected during an audit are identified and communicated to those responsible for corrective action, and resolution. | No match |  |
| **7.2.7.3.1** |  | **Process implementation** | No match |  |
| **7.2.7.3.2** |  | **Software audit** | No match |  |
| **7.2.8** | **Software Problem Resolution Process** | **7.2.8.1 Purpose**  The purpose of the Software Problem Resolution Process is to ensure that all discovered problems are identified, analyzed, managed and controlled to resolution.  **7.2.8.2 Outcomes**  As a result of successful implementation of the Software Problem Resolution Process:  a) a problem management strategy is developed;  b) problems are recorded, identified and classified;  c) problems are analyzed and assessed to identify acceptable solution(s);  d) problem resolution is implemented;  e) problems are tracked to closure; and  f) The status of all problems reported is known.  NOTE The Software Problem Resolution Process could be used or easily adapted to manage, track and control software change requests. | **Daily Scrum Meeting**  There are several practices in our process that make up a good software problem resolution process.  Daily Scrum Meetings allows team members to bring up problems that they encounter. When a problem is arise they can be put onto the Sprint Backlog which is where they are recorded. These problems should be corrected immediately. The problem should be analyzed and corrected by any developer. When the problems is corrected, it would be noted on the sprint backlog.  Testing makes it easier to discover these problems. When problems are found during testing, they are also put on to the sprint backlog. |  |
| **7.2.8.3.1** |  | **Process implementation** | No match |  |
| **7.2.8.3.2** |  | **Problem resolution** | A developer chooses which tasks (including outlying problems) to do on the sprint backlog. It is up to the developer to decide which is the best option to tackle the problem. They are allowed to ask for advice. When the problem is fixed, it is stated in the sprint backlog. |  |
| **7.3** | **Software Reuse Processes** |  |  |  |
| **7.3.1** | **Domain Engineering Process** | **7.3.1.1 Purpose**  The purpose of the Domain Engineering Process is to develop and maintain domain models, domain architectures and assets for the domain.  **7.3.1.2 Outcomes**  As a result of successful implementation of the Domain Engineering Process:  a) the representation forms for the domain models and the domain architectures are selected;  b) the boundaries of the domain and its relationships to other domains are established;  c) a domain model that captures the essential common and different features, capabilities, concepts, and  functions in the domain are developed;  d) a domain architecture describing the family of systems within the domain, including their commonalities and variabilities, is developed;  e) assets belonging to the domain are specified;  f) assets belonging to the domain are acquired or developed and maintained throughout their life cycles; and  g) The domain models and architectures are maintained throughout their life cycles. | No match |  |
| **7.3.1.3.1** |  | **Process implementation** | No match |  |
| **7.3.1.3.2** |  | **Domain analysis** | No match |  |
| **7.3.1.3.3** |  | **Domain design** | No match |  |
| **7.3.1.3.4** |  | **Asset provision** | No match |  |
| **7.3.1.3.5** |  | **Asset maintenance** | No match |  |
| **7.3.2** | **Reuse Asset Management Process** | **7.3.2.1 Purpose**  The purpose of the Reuse Asset Management Process is to manage the life of reusable assets from conception to retirement.  **7.3.2.2 Outcomes**  As a result of successful implementation of the Reuse Asset Management Process:  a) an asset management strategy is documented;  b) an asset classification scheme is established;  c) criteria for asset acceptance, certification and retirement are defined;  d) an asset storage and retrieval mechanism is operated;  e) the use of assets is recorded;  f) changes to the assets are controlled, and  g) Users of assets are notified of problems detected, modifications made, new versions created and deletion of assets from the storage and retrieval mechanism. | No match |  |
| **7.3.2.3.1** |  | **Process implementation** | No match |  |
| **7.3.2.3.2** |  | **Asset storage and retrieval definition** | No match |  |
| **7.3.2.3.3** |  | **Asset management and control** | No match |  |
| **7.3.3** | **Reuse Program Management Process** | **7.3.3.1 Purpose**  The purpose of the Reuse Program Management Process is to plan, establish, manage, control, and monitor an organization’s reuse program and to systematically exploit reuse opportunities.  **7.3.3.2 Outcomes**  As a result of successful implementation of Reuse Program Management Process:  a) the organization’s reuse strategy, including its purpose, scope, goals and objectives, is defined;  b) the domains for potential reuse opportunities are identified;  c) the organization’s systematic reuse capability is assessed;  d) the reuse potential of each domain is assessed;  e) reuse proposals are evaluated to ensure the reuse product is suitable for the proposed application;  f) the reuse strategy is implemented in the organization;  g) feedback, communication, and notification mechanisms that operate between affected parties are  established; and  h) The reuse program is monitored and evaluated.  NOTE The affected parties may include reuse program administrators, asset managers, domain engineers, developers, operators, and maintainers. | No match |  |
| **7.3.3.3.1** |  | **Initiation** | No match |  |
| **7.3.3.3.2** |  | **Domain identification** | No match |  |
| **7.3.3.3.3** |  | **Reuse assessment** | No match |  |
| **7.3.3.3.4** |  | **Planning** | No match |  |
| **7.3.3.3.5** |  | **Execution and Control** | No match |  |
| **7.3.3.3.6** |  | **Review and evaluation** | No match |  |
|  |  |  |  |  |

## Gap Analysis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Gaps between the Agile Process (Scrum&XP) to ISO 12207**  [Scope: only for the Software Implement Processes Group and the Software Support Processes Group] | | | | |
| **Clauses in ISO 12207** | **Processes**  **in ISO 12207** | **Purpose and Outcomes**  **[AND] Activities**  **in ISO 12207** | **Agile Process: Scrum & XP Practices** | **Gaps between your process (Scrum & XP practices) and ISO 12207** |
|  |  |  |  |  |
| **7** | **Software Specific Processes** |  |  |  |
| **7.1** | **Software Implementation Processes** |  |  |  |
| **7.1.1** | **Software Implementation Process** | **7.1.1.1 Purpose**  The purpose of the Software Implementation Process is to produce a specified system element implemented as a software product or service.  This process transforms specified behavior, interfaces and implementation constraints into actions that create a system element implemented as a software product or service, otherwise known as a "software item."  This process results in a software item that satisfies architectural design requirements through verification and stakeholder requirements through validation.  **7.1.1.2 Outcomes**  As a result of the successful implementation of the Software Implementation Process:  a) an implementation strategy is defined;  b) implementation technology constraints on the design are identified;  c) a software item is realized; and  d) A software item is packaged and stored in accordance with an agreement for its supply. |  |  |
| **7.1.1.3.1** |  | **Software implementation strategy** |  |  |
| **7.1.2** | **Software Requirements Analysis Process** | **7.1.2.1 Purpose**  The purpose of Software Requirements Analysis Process is to establish the requirements of the software elements of the system.  **7.1.2.2 Outcomes**  As a result of successful implementation of the Software Requirements Analysis Process:  a) the requirements allocated to the software elements of the system and their interfaces are defined;  b) software requirements are analyzed for correctness and testability;  c) the impact of software requirements on the operating environment are understood;  d) consistency and traceability are established between the software requirements and system  requirements;  e) prioritization for implementing the software requirements is defined;  f) the software requirements are approved and updated as needed;  g) changes to the software requirements are evaluated for cost, schedule and technical impact; and  h) The software requirements are baselined and communicated to all affected parties. | **7.1.2.1**  The software requirements Analysis Process 7.1.2 mapped to Scrum pre-game planning, sprint planning, and sprint review, as well as XP’s story cards, acceptance testing and onsite customer.  **7.1.2.2 Outcomes**  a) Requirements were gathered and documented into stories and organized into a Product Backlog.  b) The Product Backlog was developed in collaboration with the product owner for accuracy in the requirements. Stories contain acceptance criteria for testability. Requirements were also reviewed during the sprint review.  c) While the developers estimated the time required to complete a story, and during the creation of the tasks, the requirements impact to the environment was discussed.  d) No match  e) The Product Backlog was prioritized and was refined through each iteration as needed during the sprint planning.  f) After each iteration the product backlog was reviewed with the product owner and updated if needed.  g) Analysis for the amount of work was done by asking the developers to break down the user stories into tasks with estimates of each task. Changes to the requirements were evaluated for cost, schedule and technical impact as the developers estimated time for the stories.  h) The product backlog is the central requirements document and is openly accessible to the team. User stories were added to the Product Backlog before the work is done in the Sprint iterations. | a) No gap  b) No gap  c)No gap  d) Within Scrum and XP Methodologies there is no defined process for establishing traceability between software requirements and system requirements.  e) no gap  f) no gap  g) no gap  h) no gap |
| **7.1.2.3.1** |  | Software requirements analysis | **7.1.2.3.1.1**  Activities from Scrum and XP such as creating user stories, developing and maintaining a product backlog, developing prototypes and exploratory design and ensuring fusibility were conducted in order to gather and analyze software requirements. User stories captured functional requirements and the acceptance criteria captured various other characteristics such as performance requirements and security requirements. The scrum team refined the user stories and discussed what needed to be developed. User stories were maintained in the Product Backlog and were owned by the Product Owner. The user stories were prioritized updated during development. | **7.1.2.3.1.1**  Scrum and XP Methodologies do not specify a specific activity for documenting data definition and database requirement although these details might be captured as part of a story.  User documentation is not an activity within Scrum or XP concepts.  User operation and execution requirements are not defined as an activity or work product within Scrum or XP Concepts.  User maintenance requirements are not defined as an activity or work product within Scrum or XP concepts. |
| **7.1.3** | **Software Architectural Design Process** | **7.1.3.1 Purpose**  The purpose of the Software Architectural Design Process is to provide a design for the software that implements and can be verified against the requirements.  **7.1.3.2 Outcomes**  As a result of successful implementation of the Software Architectural Design Process:  a) a software architectural design is developed and baselined that describes the software items that will  implement the software requirements;  b) internal and external interfaces of each software item are defined; and  c) Consistency and traceability are established between software requirements and software design. |  |  |
| **7.1.3.3.1** |  | **Software architectural design** |  |  |
| **7.1.4** | **Software Detailed Design Process** | **7.1.4.1 Purpose**  The purpose of the Software Detailed Design Process is to provide a design for the software that implements and can be verified against the requirements and the software architecture and is sufficiently detailed to permit coding and testing.  **7.1.4.2 Outcomes**  As a result of successful implementation of the Software Detailed Design Process:  a) a detailed design of each software component, describing the software units to be built, is developed;  b) external interfaces of each software unit are defined; and  c) Consistency and traceability are established between the detailed design and the requirements and architectural design. |  |  |
| **7.1.4.3.1** |  | **Software detailed design** |  |  |
| **7.1.5** | **Software Construction Process** | **7.1.5.1 Purpose**  The purpose of the Software Construction Process is to produce executable software units that properly reflect the software design.  **7.1.5.2 Outcomes**  As a result of successful implementation of Software Construction Process:  a) verification criteria are defined for all software units against their requirements;  b) software units defined by the design are produced;  c) consistency and traceability are established between software units and requirements and design; and  d) Verification of the software units against the requirements and the design is accomplished. | **Development Phase Iteration/Sprint**  The software construction process maps to the development phase in both scrum and XP. Development was performed within iterations.  a) Acceptance criteria were documented in the user stories.  b) Software was developed per story and was designed to meet the stories acceptance criteria.  c) Software units were partially traceable to user requirements through the Product Backlog. Coding standards in XP provided consistency and allowed for easier refactoring.  d) Unit tests were performed in order to verify acceptance criteria. | a) no gap  b) no gap  c) Traceability between software units, requirements and design are not fully covered in XP and Scrum.  d) no gap |
| **7.1.5.3.1** |  | **Software construction**  7.1.5.3.1 Software construction. For each software item (or configuration item, if identified) this activity  consists of the following tasks:  7.1.5.3.1.1 The implementer shall develop and document the following:  a) Each software unit and database  b) Test procedures and data for testing each software unit and database.  7.1.5.3.1.2 The implementer shall test each software unit and database ensuring that it satisfies its requirements. The test results shall be documented.  7.1.5.3.1.3 The implementer shall update the user documentation as necessary.  7.1.5.3.1.4 The implementer shall update the test requirements and the schedule for Software Integration.  7.1.5.3.1.5 The implementer shall evaluate software code and test results considering the criteria listed below. The results of the evaluations shall be documented.  a) Traceability to the requirements and design of the software item.  b) External consistency with the requirements and design of the software item.  c) Internal consistency between unit requirements.  d) Test coverage of units.  e) Appropriateness of coding methods and standards used.  f) Feasibility of software integration and testing.  g) Feasibility of operation and maintenance. | **XP Pair Programming**  Software was developed using the XP Pair programming concepts.  XP - Software unit tests are written and then the code is developed. The other developers understand the code by reviewing the tests.  7.1.5.3.1.1  Software and unit tests were developed.  7.1.5.3.1.2  Unit tests were performed in order to ensure that requirements were met.  7.1.5.3.1.3  No match  7.1.5.3.1.4  XP- Continuous integration was performed after unit tests were successful.  7.1.5.3.1.5  Evaluation of code was conducted at the time of development as the code was written by a pair of developers. Unit tests were developed as the code was developed. User stories partially provide traceability from the software item to the requirements and design. Pair programming and code standards were used to ensure consistency within the code construction. A constant review of the user stories and product backlog by the scrum team was used to ensure consistency within the user requirements. XP continuous integration was used to review feasibility of software integration. | More documentation is required to be part of the specifications for each software unit and database along with testing procedures.  The implementer must test the code by evaluation the code is consistent with the requirements and overall design of the software system.  7.1.5.3.1.1  Software units and database were not documented.  7.1.5.3.1.2  Test results were not documented.  7.1.5.3.1.3  User documentation was not developed.  7.1.5.3.1.4  no gap  7.1.5.3.1.5  Test results and evaluation of software code was not documented. Traceability from software item to the requirement and design is not fully documented. Feasibility of operation and maintenance was not covered. |
| **7.1.6** | **Software Integration Process** | **7.1.6.1 Purpose**  The purpose of the Software Integration Process is to combine the software units and software components, producing integrated software items, consistent with the software design, that demonstrate that the functional and non-functional software requirements are satisfied on an equivalent or complete operational platform.  **7.1.6.2 Outcomes**  As a result of successful implementation of the Software Integration Process:  a) an integration strategy is developed for software units consistent with the software design and the prioritized software requirements;  b) verification criteria for software items are developed that ensure compliance with the software  requirements allocated to the items;  c) software items are verified using the defined criteria;  d) software items defined by the integration strategy are produced;  e) results of integration testing are recorded;  f) consistency and traceability are established between software design and software items; and  g) a regression strategy is developed and applied for re-verifying software items when a change in software units (including associated requirements, design and code) occur. |  |  |
| **7.1.6.3.1** |  | **Software integration** |  |  |
| **7.1.7** | **Software Qualification Testing Process** | **7.1.7.1 Purpose**  The purpose of the Software Qualification Testing Process is to confirm that the integrated software product meets its defined requirements.  **7.1.7.2 Outcomes**  As a result of successful implementation of the Software Qualification Testing Process:  a) criteria for the integrated software is developed that demonstrates compliance with the software  requirements;  b) integrated software is verified using the defined criteria;  c) test results are recorded; and  d) A regression strategy is developed and applied for re-testing the integrated software when a change in software items is made. |  |  |
| **7.1.7.3.1** |  | **Software qualification testing** |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **7.2** | **Software Support Processes** |  |  |  |
| **7.2.1** | **Software Documentation Management Process** | **7.2.1.1 Purpose**  The purpose of the Software Documentation Management Process is to develop and maintain the recorded software information produced by a process.  **7.2.1.2 Outcomes**  As a result of successful implementation of the Software Documentation Management Process:  a) a strategy identifying the documentation to be produced during the life cycle of the software product or  service is developed;  b) the standards to be applied for the development of the software documentation are identified;  c) documentation to be produced by the process or project is identified;  d) the content and purpose of all documentation is specified, reviewed and approved;  e) documentation is developed and made available in accordance with identified standards; and  f) Documentation is maintained in accordance with defined criteria. |  |  |
| **7.2.1.3.1** |  | **Process implementation** |  |  |
| **7.2.1.3.2** |  | **Design and development** |  |  |
| **7.2.1.3.3** |  | **Production** |  |  |
| **7.2.1.3.4** |  | **Maintenance** |  |  |
| **7.2.2** | **Software Configuration Management Process** | **7.2.2.1 Purpose**  The purpose of the Software Configuration Management Process is to establish and maintain the integrity of the software items of a process or project and make them available to concerned parties.  **7.2.2.2 Outcomes**  As a result of successful implementation of the Software Configuration Management Process:  a) a software configuration management strategy is developed;  b) items generated by the process or project are identified, defined and baselined;  c) modifications and releases of the items are controlled;  d) modifications and releases are made available to affected parties;  e) the status of the items and modifications are recorded and reported;  f) the completeness and consistency of the items is ensured; and  g) The storage, handling and delivery of the items are controlled. | Scrum and XP process were mapped to software configuration management through the following processes: continuous integration, pair programming, and collective ownership, release planning.  a) Partially mapped to continuous integration concept from XP. Changes were controlled by only allowing one pair to check in at a time. There was a dedicated computer that stored the source code.  b) No match  c) Releases were completed at end of each iteration. Change requests for modifications are reviewed during the sprint planning.  d) The releases were delivered to the stakeholders as scheduled within the release plan.  e) The status of the stories in the iteration was tracked using a burndown chart. The status of the release was given during the sprint review at the end of each iteration.  f) Stories were compared against acceptance criteria to ensure completeness. Consistency was ensured by developing the code using pair programming and by following the standards.  g) No match | a) Scrum and XP do not have a process to develop a plan to carry out the XP process for configuration management strategy.  b) Scrum and XP does not have a process to document the baseline and the version referenced.  c) no gap  d) no gap  e) no gap  f) no gap  g) Scrum and XP do not have a process for storage, handling and delivery of the items. |
| **7.2.2.3.1** |  | **Process implementation** | No match | Scrum and XP do not have a process for developing a configuration management plan. |
| **7.2.2.3.2** |  | **Configuration identification** | No match | Scrum and XP do not include version references in the documentation for the software items. |
| **7.2.2.3.3** |  | **Configuration control** | No match | A clear audit trail does not exist that enables the team to trace modified software data. |
| **7.2.2.3.4** |  | **Configuration status accounting** | The status was tracked using the burndown chart and XP product velocity and were reported to the stakeholders during the sprint review. | Scrum and XP does not contain item versions or release identifiers. |
| **7.2.2.3.5** |  | **Configuration evaluation** | Developers checked for completeness of the requirements upon completing the stories. Acceptance testing was conducted to ensure that the software meets the requirements. | No gap. |
| **7.2.2.3.6** |  | **Release management and delivery** | The code is maintained for the releases as a master copy. | Once the development phase is over, Scrum and XP do not have a process for maintaining the documentation. |
| **7.2.3** | **Software Quality Assurance Process** | **7.2.3.1 Purpose**  The purpose of the Software Quality Assurance Process is to provide assurance that work products and processes comply with predefined provisions and plans.  **7.2.3.2 Outcomes**  As a result of successful implementation of the Software Quality Assurance Process:  a) a strategy for conducting quality assurance is developed;  b) evidence of software quality assurance is produced and maintained;  c) problems and/or non-conformance with requirements are identified and recorded; and  d) Adherence of products, processes and activities to the applicable standards, procedures and requirements are verified. | Pair programming, collective code ownership and acceptance testing concepts from XP partially map to quality assurance but do not fully encompass a quality assurance process.  Pair programming and collective code ownership help ensure that code is developed to meet the team standards. Acceptance testing ensures that software is written to meet the requirements. Bug items are documented if a requirement is not met correctly.  a) Partially maps to Scrum retrospective meetings which were conducted after each iteration. The iteration was reviewed to identify possible improvements in the process.  b) No match  c) Partially mapped to Scrum retrospective. Issues in the process were identified and recorded.  d) Partially maps to Scrum Master as the Scrum Master verified that the process is being followed correctly. Scrum Master should be trained. | a) XP and Scrum do not have a process for developing a strategy for quality assurance.  b) Quality assurance is not documented through XP or Scrum.  c) Non-conformance are not recorded.  d) Standards are not verified within Scrum or XP. |
| **7.2.3.3.1** |  | **Process implementation** | No match | A Quality Assurance process is not implemented or followed. |
| **7.2.3.3.2** |  | **Product assurance** | Pair programming, collective code ownership, refactoring, continuous integration and test driven development partially map and were used to ensure the quality of the product. | Product quality assurance is not documented with Scrum and XP. |
| **7.2.3.3.3** |  | **Process assurance** | Scrum Master verifies that the process is being followed during the software development life cycle. | The team’s process is not measured against Scrum and XP process. |
| **7.2.3.3.4** |  | **Assurance of quality systems** | No match | Scrum and XP does not assure that process is in accordance with ISO 9001. |
| **7.2.4** | **Software Verification Process** | **7.2.4.1 Purpose**  The purpose of the Software Verification Process is to confirm that each software work product and/or service of a process or project properly reflects the specified requirements.  **7.2.4.2 Outcomes**  As a result of successful implementation of the Software Verification Process:  a) a verification strategy is developed and implemented;  b) criteria for verification of all required software work products is identified;  c) required verification activities are performed;  d) defects are identified and recorded; and  e) Results of the verification activities are made available to the customer and other involved parties. |  |  |
| **7.2.4.3.1** |  | **Process implementation** |  |  |
| **7.2.4.3.2** |  | **Verification** |  |  |
| **7.2.5** | **Software Validation Process** | **7.2.5.1 Purpose**  The purpose of the Software Validation Process is to confirm that the requirements for a specific intended use of the software work product are fulfilled.  **7.2.5.2 Outcomes**  As a result of successful implementation of the Software Validation Process:  a) a validation strategy is developed and implemented;  b) criteria for validation of all required work products are identified;  c) required validation activities are performed;  d) problems are identified and recorded;  e) evidence is provided that the software work products as developed are suitable for their intended use;  and  f) Results of the validation activities are made available to the customer and other involved parties. |  |  |
| **7.2.5.3.1** |  | **Process implementation** |  |  |
| **7.2.5.3.2** |  | **Validation** |  |  |
| **7.2.6** | **Software Review Process** | **7.2.6.1 Purpose**  The purpose of the Software Review Process is to maintain a common understanding with the stakeholders of the progress against the objectives of the agreement and what should be done to help ensure development of a product that satisfies the stakeholders. Software reviews are at both project management and technical levels and are held throughout the life of the project.  **7.2.6.2 Outcomes**  As a result of successful implementation of the Software Review Process:  a) management and technical reviews are held based on the needs of the project;  b) the status and products of an activity of a process are evaluated through review activities;  c) review results are made known to all affected parties;  d) action items resulting from reviews are tracked to closure; and  e) Risks and problems are identified and recorded. |  |  |
| **7.2.6.3.1** |  | **Process implementation** |  |  |
| **7.2.6.3.2** |  | **Project Management Reviews** |  |  |
| **7.2.6.3.3** |  | **Technical Reviews** |  |  |
| **7.2.7** | **Software Audit Process** | **7.2.7.1 Purpose**  The purpose of the Software Audit Process is to independently determine compliance of selected products and processes with the requirements, plans and agreement, as appropriate.  **7.2.7.2 Outcomes**  As a result of successful implementation of the Software Audit Process:  a) an audit strategy is developed and implemented;  b) compliance of selected software work products and/or services or processes with requirements, plans  and agreement is determined according to the audit strategy;  c) audits are conducted by an appropriate independent party; and  d) Problems detected during an audit are identified and communicated to those responsible for corrective action, and resolution. |  |  |
| **7.2.7.3.1** |  | **Process implementation** |  |  |
| **7.2.7.3.2** |  | **Software audit** |  |  |
| **7.2.8** | **Software Problem Resolution Process** | **7.2.8.1 Purpose**  The purpose of the Software Problem Resolution Process is to ensure that all discovered problems are identified, analyzed, managed and controlled to resolution.  **7.2.8.2 Outcomes**  As a result of successful implementation of the Software Problem Resolution Process:  a) a problem management strategy is developed;  b) problems are recorded, identified and classified;  c) problems are analyzed and assessed to identify acceptable solution(s);  d) problem resolution is implemented;  e) problems are tracked to closure; and  f) The status of all problems reported is known.  NOTE The Software Problem Resolution Process could be used or easily adapted to manage, track and control software change requests. |  |  |
| **7.2.8.3.1** |  | **Process implementation** |  |  |
| **7.2.8.3.2** |  | **Problem resolution** |  |  |

## Rating

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Rating Process Attributes (PA) and Capability Levels (CL)**  [Scope: both the Software Implement Processes Group and the SW Support Processes Group]  (for 2 processes [SW Requirements Analysis Process, SW Construction Process] in the *Software Implementation Processes Group,* and  2 processes [SW Configuration Management Process, Software Quality Assurance Process] in the *Software Support Processes Group*)  [Target Profile: CL 3]  **Process Attributes Rating [N/P/L/F] and Capability Rating [CL0 – CL5]\*:**  \*See the ISO 15504 lecture notes for details (especially slide#41-#70 for rating). | | | | | | |
| **Clauses in ISO 12207** | **Processes**  **in ISO 12207** | **Process Attributes (PAs) in ISO 12207** | **Required PA Rating for CL3** | **Process Attribute Rating** | **Gaps Found in Rating** | **CL Rating** |
| **7.1** | **Software Implementation Processes** |  |  |  |  |  |
| **7.1.1** | **Software Implementation Process** | **PA 3.2 Process Deployment** |  |  |  |  |
| **PA 3.1 Process Definition** |  |  |  |
| **PA 2.2 Work Product Management** |  |  |  |
| **PA 2.1 Performance management** |  |  |  |
| **PA 1.1 Process Performance** |  |  |  |
| **7.1.2** | **Software Requirements Analysis Process** | **PA 3.2 Process Deployment** | L or F | L | More than 50% performed.  No data is collected to demonstrate the suitability of the process. | CL3 |
| **PA 3.1 Process Definition** | L or F | L | More than 50% of the process attributes were met. |
| **PA 2.2 Work Product Management** | F | F | More than 85% of the process attributes were met. |
| **PA 2.1 Performance management** | F | F | More than 85% of the process attributes were met. |
| **PA 1.1 Process Performance** | F | F | More than 85% of the process attributes were met. |
| **7.1.3** | **Software Architectural Design Process** | **PA 3.2 Process Deployment** |  |  |  |  |
| **PA 3.1 Process Definition** |  |  |  |
| **PA 2.2 Work Product Management** |  |  |  |
| **PA 2.1 Performance management** |  |  |  |
| **PA 1.1 Process Performance** |  |  |  |
| **7.1.4** | **Software Detailed Design Process** | **PA 3.2 Process Deployment** |  |  |  |  |
| **PA 3.1 Process Definition** |  |  |  |
| **PA 2.2 Work Product Management** |  |  |  |
| **PA 2.1 Performance management** |  |  |  |
| **PA 1.1 Process Performance** |  |  |  |
| **7.1.5** | **Software Construction Process** | **PA 3.2 Process Deployment** | L or F | L | More than 50% of the process attributes were met. | CL3 |
| **PA 3.1 Process Definition** | L or F | L | More than 50% of the process attributes were met. |
| **PA 2.2 Work Product Management** | F | F | More than 85% of the process attributes were met. |
| **PA 2.1 Performance management** | F | F | More than 85% of the process attributes were met. |
| **PA 1.1 Process Performance** | F | F | More than 85% of the process attributes were met. |
| **7.1.6** | **Software Integration Process** | **PA 3.2 Process Deployment** |  |  |  |  |
| **PA 3.1 Process Definition** |  |  |  |
| **PA 2.2 Work Product Management** |  |  |  |
| **PA 2.1 Performance management** |  |  |  |
| **PA 1.1 Process Performance** |  |  |  |
| **7.1.7** | **Software Qualification Testing Process** | **PA 3.2 Process Deployment** |  |  |  |  |
| **PA 3.1 Process Definition** |  |  |  |
| **PA 2.2 Work Product Management** | F |  |  |
| **PA 2.1 Performance management** |  |  |  |
| **PA 1.1 Process Performance** |  |  |  |
|  |  |  |  |  |  |  |
| 7.2 | **Software Support Processes** |  |  |  |  |  |
| 7.2.2 | **Software Configuration Management Process** | **PA 3.2 Process Deployment** | L or F | N | 0 to 15 % achievement | CL0 |
| **PA 3.1 Process Definition** | L or F | N | 0 to 15 % achievement |
| **PA 2.2 Work Product Management** | F | N | 0 to 15 % achievement |
| **PA 2.1 Performance management** | F | P | Less than 50% of the process attributes were met. |
| **PA 1.1 Process Performance** | F | P | Less than 50% of the process attributes were met. |
| 7.2.3 | **Software Quality Assurance Process** | **PA 3.2 Process Deployment** | L or F | N | 0 to 15 % achievement | CL0 |
| **PA 3.1 Process Definition** | L or F | N | 0 to 15 % achievement |
| **PA 2.2 Work Product Management** | F | N | 0 to 15 % achievement |
| **PA 2.1 Performance management** | F | P | Less than 50% of the process attributes were met. |
| **PA 1.1 Process Performance** | F | P | Less than 50% of the process attributes were met. |

## Capability Determination

After the assessment, the Capability Level was determined for these processes.

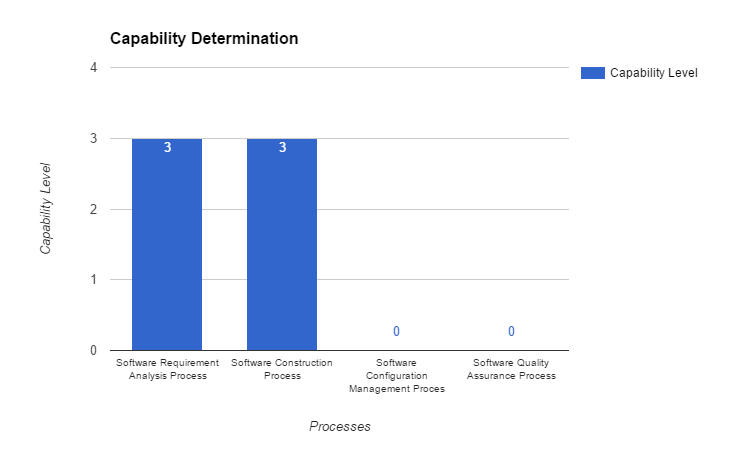


Figure : Capability Determination

|  |  |
| --- | --- |
| **Process Attributes** | **Capability Level** |
| Process Optimization (5.2) | 5 |
| Process Improvement (5.1) |
| Process Control (4.2) | 4 |
| Process Measurement (4.1) |
| Process Deployment (3.2) | 3 |
| Process Definition (3.1) |
| Work Product Performance (2.2) | 2 |
| Performance Management (2.1) |
| Process Performance (1.1) | 1 |
|  | 0 |

Figure : Process Attributes

## Target and Assessed Process Profiles

After the assessment, the target and assessed process profiles showed which assessed processes did not meet the target Process Attribute Rating. The Software Requirement Analysis Process and the Software Construction process both meet their target while the other two processes failed to meet the target by only partially achieving the Process Performance and Performance Management attributes.

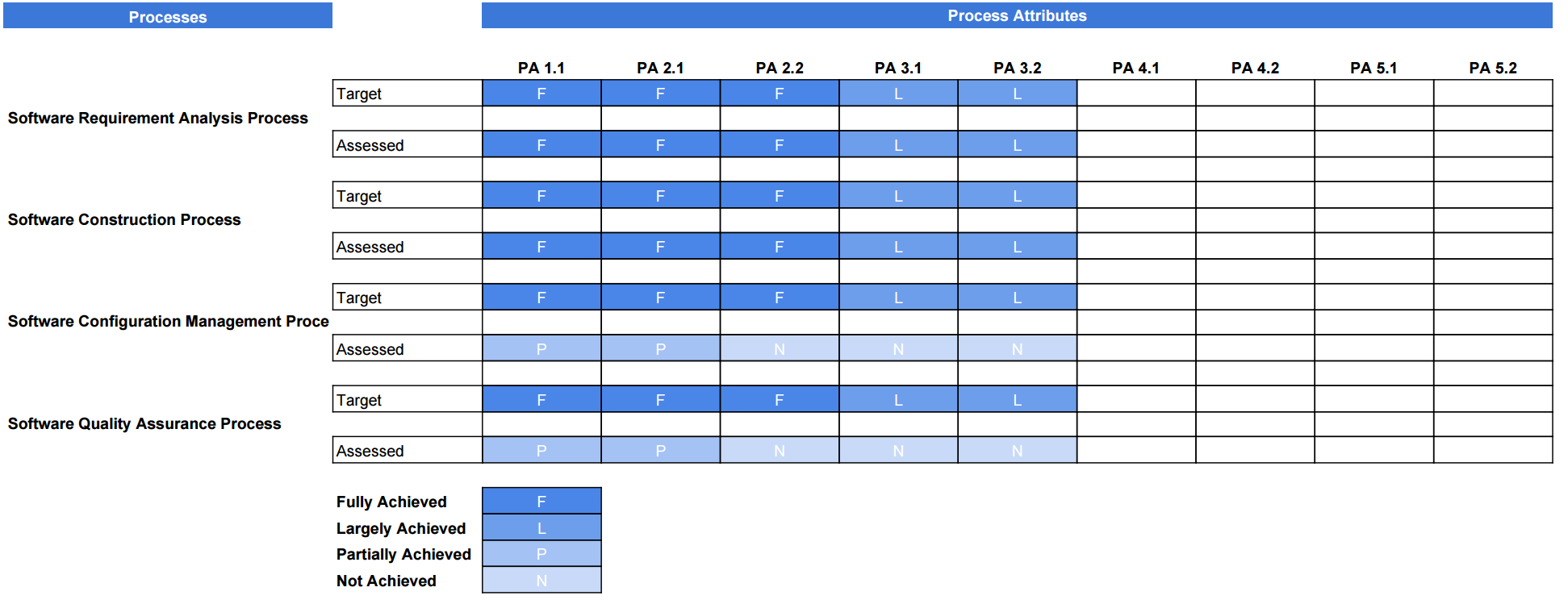


Figure : Target versus Assessed Process Profiles

# 

# Step 4 – Develop action plan

We developed an action plan to improve our process. We began by identifying improvement areas, analyzing assessment strengths and weaknesses, reviewing organizational improvement objectives, analyze effective measurements, define detailed improvement objectives and set targets, and derive action plan. These activities are detailed further below.

## Identify improvement areas

We identified the process improvement areas to build the action plan based on the ratings for each process within the scope of the assessment. Our target capability level was CL3. Based on our determination we can provide the recommended actions to alter our existing process to improve our capability.

The table below shows the Capability Level that was met for the following Software Specific Processes.

|  |  |
| --- | --- |
| Process | Capability Level (CL) |
| 7.1.2 Software Requirement Analysis Process | 3 |
| 7.1.5 Software Construction Process | 3 |
| 7.2.2 Software Configuration Management Process | 0 |
| 7.2.3 Software Quality Assurance Process | 0 |

Since 7.1.2 Software Requirement Analysis Process and 7.1.5 Software Construction Process already met the target CL3, 7.2.2 Software Configuration Management Process and 7.2.3 Software Quality Assurance Process were identified as the improvement areas.

## Analyze assessment strengths and weaknesses

From our mapping, 7.1.2 Software Requirements Analysis Process and 7.1.5 Software Construction Process reached the target Capability Level 3, these processes were identified as our strengths whereas 7.2.2 Software Configuration Management Process and 7.2.3 Software Quality Assurance Process were Capability Level 0. 7.2.2 and 7.2.3 had low Process Attribute Ratings compared to its target PAR. These low process attribute ratings indicated that the processes were missing key practices.

|  |  |
| --- | --- |
| **Strengths** | **Weaknesses** |
| 7.1.2 Software Requirement Analysis Process | 7.2.2 Software Configuration Management Process |
| 7.1.5 Software Construction Process | 7.2.3 Software Quality Assurance Process |

## 

## Review organizational improvement objectives

The organization in Step One set organizational improvement objectives. These objectives can be matched with one or multiple processes. With this mapping, the organization can determine rankings for improving each process based off how they will achieve the organization’s improvement objectives. Below, a table maps processes to the improvement objectives that were stated in Step One.

|  |  |
| --- | --- |
| Process | Organizational Improvement Objective |
| 7.1.2 Software Requirement Analysis Process | * Ensure products adhere to requirements that meet stakeholder’s objectives * Improve development efficiency - less rewrite through better requirements analysis * Enable predictive instead of reactive changes to software or systems |
| 7.1.5 Software Construction Process | * Deliver products within budget on time according to schedule * Establish measurement processes with a strategy for continuous process improvement * Identify risks to meeting project objectives early * Identify process improvement areas to improve in order to deliver higher quality products and increase organizational capability * Preserve the integrity of a SW design * Build on organization strengths and improve areas of weakness and creation of standards for development processes to increase effectiveness * The plan that is developed can be reused with projects with similar scope and produce consistent results |
| 7.2.2 Software Configuration Management Process | * Ensure adaptive Life Cycle processes are utilized to be responsive to changing requirements * Monitor progress to goals and avoid risky last minute changes * Preserve the integrity of a SW design * Improve change management * Enable predictive instead of reactive changes to software or systems * Maintain processes after improvements have been established |
| 7.2.3 Software Quality Assurance Process | * Increase overall product quality * Ensure products adhere to requirements that meet stakeholder’s objectives * Preserve the integrity of a SW design * Maintain processes after improvements have been established |

From the table above, the Software Construction Process (CL 3) addresses the most improvement objectives with seven followed by the Software Configuration Management Process (CL 0) with addressing six objectives. The Software Quality Assurance Process (CL 0) addresses four while the Software Requirement Analysis Process (CL 3) addresses the least amount of improvement objectives with three.

With improvement objectives in mind along with the current capability level of each of these processes it may be best to prioritize improving these process as such:

|  |  |
| --- | --- |
| Process | Priority Rank |
| 7.2.2 Software Configuration Management Process | 1 |
| 7.1.5 Software Construction Process | 2 |
| 7.2.3 Software Quality Assurance Process | 3 |
| 7.1.2 Software Requirement Analysis Process | 4 |

7.2.2 Software Configuration Management Process was prioritized over 7.2.3 Software Quality Assurance Process due to our organization’s business goals. In addition, software configuration management maintains the integrity of code whereas quality code can be overwritten and lost without a proper software configuration management process. Therefore, from our organization’s business goals, we inferred software quality assurance process cannot be improved without a solid software configuration management process.

## Analyze effectiveness measurements

Our team’s analysis of its need for improvement with Scrum and XP came from two areas within our software configuration management; one from change management and one from testing. With change management we used pair programming, which allowed for our team to actively look at what each team member was contributing to the code and thus give input on how the change could be bettered or adjusted. Secondly, the use of acceptance testing after each iterative step in the development process allowed our team to actively see what was at fault with the current code and what part of the code was needed to be changed in order to assure a quality product. These methods however, are not as structured or effective as ISO 15504 methods of process management is.

With ISO 15504, each area of the process was mapped out and the analysis was quantitatively constructed. With the previous form of improvement analysis, changes were very much subjected to the opinion of the team, per line of code, rather than seeing which part of the entire area of development needs work. With the new system we looked at broad areas of the process and numbered them according to our strengths and weaknesses. Then from there we could see where we needed to improve. With each iteration of improvement the entirety of the process becomes stronger therefore, the entirety of development gets better, rather than a few lines of improvement at a time.

## Define detailed improvement objectives and set targets

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Improvement Objective | Measurement of Completeness | Target Completion |
| **7.2.2 Software Configuration Management Process** |  |  | **CL0 to CL1** |
|  | Create and fully document a plan for Configuration Management | When a plan has been created and is used for next development phase |  |
|  | Have a change review board that reviews desired changes | When a board has been hired/selected and is actively reviewing changes |  |
|  | Every developed piece of code has a specified documents to it. | List out each piece of code without documentation. When the list is completed. |  |
|  | Create an audit trail to track modified code | When changes in code are being actively tracked |  |
| **7.2.3 Software Quality Assurance Process** |  |  | **CL0 to CL1** |
|  | Create a QA team | QA team selected/appointed |  |
|  | Define measures to effectively record and analyze development lifecycle processes effectively | Measures are approved by the QA Team |  |
|  | Develop a plan to support the Measurement and Analysis process | Plan is documented and approved |  |
|  | Collecting and analyzing data for QA | Database of measures is maintained by the QA team |  |
|  | Report key findings to management along with process improvement recommendations. | Conduct periodic meetings to report findings |  |

## Derive action plan

|  |  |
| --- | --- |
| 7.2.2 Software Configuration Management Process | |
| **Improvement** | Create and fully document a plan for Configuration Management |
| Actions | 1. Document SCM Organization 2. Document SCM Methods 3. Document SCM Procedures 4. Document SCM Implementation |
| Responsibilities | Development Team |
| Cost, Benefits, Schedule | Cost: 8 hours of developer pay  Benefits: Code will not get lost  Schedule: 2, 4 hour sessions |
| Risk | Lose 8 hours of development time if no plan is created |
|  |  |
| **Improvement** | Have a change review board that reviews desired changes |
| Actions | 1. Hire/select CRB 2. create a system of documentation for the CRB to review changes |
| Responsibilities | Managers, Development team |
| Cost, Benefits, Schedule | Cost: 3 hours of work for development team to create the system; salary for the newly created CRB  Benefits: Changes are scrutinized and only beneficial changes will be approved.  Schedule: 1, 3 hour development session. 1 meeting a week to discuss changes |
| Risk | CRB may not do the job correctly, and since changes are reviewed with them and no one else, bad changes could be easily approved. |
|  |  |
| **Improvement** | Every developed piece of code has a specified documents to it. |
| Actions | 1. Create a system for documentation within the code, i.e. comments, .txt, Javadoc. 2. Document past code, starting with code pertinent to current customer then expand over time 3. Ensure code is continuously documented |
| Responsibilities | Development team |
| Cost, Benefits, Schedule | Cost: The hours required to go into past code, can’t be determined  Benefits: developers will have the ability to understand the reasoning for implementation of specific code when others do not know.  Schedule: 2, 4 hour sessions of important code. Continuous documentation with current code. When time permits, have a 1 hour session of old code documentation. |
| Risk | None, documentation of code has no risk to it other than time restraints. |
|  |  |
| **Improvement** | Create an audit trail to track modified code |
| Actions | 1. Create a system that will track modified code. 2. begin tracking modified code |
| Responsibilities | Development team |
| Cost, Benefits, Schedule | Cost: 3 Hours of development salary.  Benefits: Being able to see what code was changed.  Schedule: 1, 3 hour session |
| Risk | Wasted time if it does not work or the developers cannot create an audit trail |
|  | |
| **7.2.3 Software Quality Assurance Process** | |
| **Improvement** | Create a QA team |
| Actions | 1. Hire QA Team |
| Responsibilities | Management |
| Cost, Benefits, Schedule | Cost: Salary of the QA team  Benefit: Having a QA team to assure software quality  Schedule: 1 hiring cycle |
| Risk | We could find no qualified QA team and the time used on the hiring cycle is wasted |
|  |  |
| **Improvement** | Define measures to effectively record and analyze development lifecycle processes effectively |
| Actions | 1. Create measures 2. create the system to record 3. meet and analyze the development process |
| Responsibilities | Development team, QA team |
| Cost, Benefits, Schedule | Cost: 1 hour of time to analyze per two weeks, 1 hour to create measures and system.  Benefits: Having a stem for the QA team to analyze the development lifecycle as well as if the process is effective  Schedule: 1 hour meeting a week for analysis. (This will be in combination with the 2 hour meeting in “Collecting and analyzing data for QA”) |
| Risk | The measures could lead to no improvement in the process nor in the lifecycle. Thus wasting money and time for which QA could have spent on assuring quality |
|  |  |
| **Improvement** | Develop a plan to support the Measurement and Analysis process |
| Actions | 1. Develop the plan including: Development, Product, Acceptance, Usage, and Repair. |
| Responsibilities | QA team |
| Cost, Benefits, Schedule | Cost: Pay time to develop the plan, 1 hour.  Benefit: There will be a way for the QA team to fully analyze what will allow for the greatest representation of user needs.  Schedule: 1 hour planning time |
| Risk | A waste of 1 hour for the planning. |
|  |  |
| **Improvement** | Collecting and analyzing data for QA |
| Actions | 1. Create a document that will be used to collect data for QA 2. QA analyzes the data collected once per week |
| Responsibilities | Development team, QA team |
| Cost, Benefits, Schedule | Cost: 1 hour of development time; 2 hours each week for QA analysis.  Benefits: Having a comprehensive collection of data for the QA team to better their assurance process.  Schedule: 1 hour of development time. 2 hours each week set aside for QA team to meet. |
| Risk | The data could not improve any of the QA process and the time wasted on the 2 hours of meetings per week could hinder the QA process |

# Step 5 – Implement improvements

Not included in this homework

# Step 6 – Confirm improvements

Not included in this homework

# Step 7 – Sustain improvements

Not included in this homework

# Step 8 – Monitor performance

Not included in this homework

# Lessons Learned

## Anthony Farina

Between the two homework assignments for CPSC 544, I have learned a tremendous amount of knowledge. Because of my lack of experience as a software engineer, I have yet to encounter many facets of the profession. I have not yet encountered an actual software development process. My experience has been haphazard at best. During homework one, it was a very refreshing experience. It taught me that there is a way to develop software in an organized way without sacrificing quickness or the ability to deal with change quickly. Agile processes seem to be the best processes for the company that I work for. I may start implementing an official process into the company in order to increase our maturity.

Homework two also taught me a great deal about “formal” software processes and process assessments. It showed me the positives and negatives of Agile and the positives and negatives of an ISO 12207 complaint process. It taught me that there are other ways to go about developing software. Another big part of homework two was performing a process assessment. This proved to be very beneficial since it taught me how to improve upon a process. This has to be a very important skill to have. It teaches you about the process while at the same time helping you improve the process by finding its holes.

In the future, I would really like to learn about more types of processes and the scenarios in which they work and don’t work. I would also like to learn which processes more mature companies use. Learning what real world companies use and how they implement processes is really intriguing to me.

## Lourdes Lopez

My experience with process definition, assessment and improvement comes from working with Scrum methodologies. I was part of a team that began adopting Scrum so I was able to see the process of implementing process. First the team began by learning about scrum and educating the business owners. One of the most notable differences was having a product owner involved in the development process. Throughout a few months the team struggled to refine and maintain process. The process was continuously examined. The next step that worked well for the team was bringing in a consultant to observe and review the team's process against scrum methodologies to see if we were following the practices correctly. After making some modifications based on the consultant's recommendation we saw notable improvement. At my current employment we do not have formal processes for the majority of our work. It is very difficult to go from using process to an organization that does not follow any formal process but it gives me the opportunity to implement process.

Both homework one and homework two have been a learning experience for me. With the first homework I learned more about scrum and also welcomed the opportunity to learn about XP. Since I had some experience with scrum I was most interested in learning about the differences and similarities between Scrum and XP and also how they can be used in combination. The second homework helped me explore process at a much broader level. By performing the ISO 15504 assessment I was able to explore all aspects of software development process beyond Scrum and XP. I learned about assessing a process and making a plan for improving the process, both of which will be very useful to me going forward in my career.

One thing that I will most likely explore farther is the software reuse processes. I have yet to be a part of a team that can successfully implement a process for reusing software and am interested in how it can be planned and implemented.

## Joanna Hang

Prior to these homework assignments, I did not have any experience with process, process definition, process assessment or process improvement. I learned that process assessment and process improvement are long-term commitments and can consume many resources, but are necessary in improving an organization’s maturity level. Overall, these two homework assignments taught me that there are many faucets to software engineering that were previously unknown to me.

On HW1, I learned about iterative development versus the traditional waterfall method. I also learned about agile practices, specifically Scrum and XP methods. I found some of these practices very useful for project management and development. For example, I found Scrum’s sprint planning and sprint review particularly helpful in setting requirements with customers because the customers were now responsible for prioritizing their requirements and began to understand that development takes time and not all requirements can be implemented at once. With XP, I enjoyed practicing pair programming as I had the opportunity to learn from other developers.

HW2 entirely opened my views with software engineering practices with ISO 12207 and ISO 15504. Prior to this HW, I had a limited view of software engineering practices by working in a small development environment. I understood that ISO 12207 may be more suitable for large companies or large government agencies like Boeing or NASA or suitable for products that are high criticality. However, I felt some of the terms were too vague in ISO 12207 and could be interpreted in multiple ways. Therefore, I wish to learn about actual examples for ISO 12207. I also wish to learn more software engineering practices that are more suitable for small technical teams.

## David Sullivan

Working on an actual development project using the Scrum and XP processes was a great learning experience because I have not had an opportunity to try using a software development process my organization. By taking different roles in the process, it helped me to see the development from different perspectives. It was useful to start with a process and build a plan to follow that helps you to clear up any misunderstandings in the requirements and work together with everyone when there are more clearly defined goals. While this project was relatively simple, we could easily coordinate on the project even without all having worked together before. The pair programming was a good way to get up to speed on the Android platform. I could see how following the agile process could increase the product value and speed up development time.

In the first half of the homework, I could easily grasp the development process by performing the Agile processes by building our custom process and following it. I found out later in Homework 2 that it was quite simple compared to the formal software processes that are defined in ISO 12207. It was confusing trying to map agile processes to the process reference model. It must be done iteratively because you may misread or not think of the mapping or gaps to other standards until you have spent a lot of time understanding going over how to do an assessment.

Even though with Agile processes you can start coding almost right away, there are drawbacks to the Agile process. Agile methods alone cannot reach higher maturity levels if you only follow those processes. However, by modifying Agile, you could achieve a higher maturity level. For example, Agile does not say you do not need to document at all, but it just not emphasize it enough if you are trying to build a highly defined process across an organization.

I am more comfortable using Agile because it is more similar to my current approach development in a small group. However, it would seem that this methodology implies that you should also include other activities from more mature software lifecycle process. With highly critical systems, this is especially true. I think the emphasis on real-time communication (with light documentation) has some risks and relies on being able to make changes easily but could run into design-related problems that might not be so easy to correct. I would be more comfortable to start with a more defined plan as well as having a little more of a detailed process to follow. The ISO 12207 process provides a more comprehensive approach that could remove risks and improve the overall process (and resulting software).

On the other hand, waterfall methods are more complex and it may be difficult to implement in small organizations. Agile is a good starting point for most organizations. Some of the processes align with the heavy standards such as ISO 12207. As the organization strives to improve the process, changes could be slowly introduced such as a Software Change Management system and a more formal process to do SQA and inspections rather than verification done by a small scrum team only. But the question is, should an Agile process even be doing these higher level activities? I think it depends on the size of the project and the requirements of the client or business goals to determine whether to use Agile development or if a more complex process model should be implemented.

I would be interesting to learn more about implementing the process improvements. In my organization we find errors too late in the process and it can create a lot of stressful situations. I would like to learn more about XP and build unit testing into my development process. I would also like to learn more about creating a software configuration management plan to control software items being produced and avoid common problems by coordinating the development activities more efficiently than we do now.

## Timothy Cioffi

The team I currently work with does not use any of the process’ I have learned about from this class. The way the development works is very amateur and lacks any long lasting improvements to the process. I see this now due what I have learned from homeworks one and two. A customer's calls for a change, a change is talked about by management, the change is sent to developers, the change is committed, and QA tests the change. There is no overall control over how the process is, and everything is very top-down driven. Learning about the Agile process and ISO software improvement has really opened my eyes to some of the flaws with this system.

In the first homework we looked at the Agile process and how it could be applied to our development. Agile allowed for quick code production and effective change management. This allowed for our team to quickly develop a working product that was able to be released in a very short time, and then consistently improve that product in a stable pattern for which quick changes and tests could be performed iteratively. While my current team does do a similar pattern to this, it does not encompass the iterative aspect for which Agile does. If my team were to look at each change as an iteration, with a series of acceptance tests built into each change before new changes were implemented, I think my team would get out better changes then it currently does.

In the second homework we looked at ISO 12207 and ISO 15504, which looked at a different way of handling the software lifecycle and process improvements. This looks at a wide range of aspects with the lifecycle and quantitatively rates each part of the process. What this does is gives the team a way of analyzing where the bulk of problems with the development process lie, giving the team an area of focus. From there the team can focus on improving an area of the process towards development rather than each change to the product, as with Agile. Overall, the process itself will improve and thus the product will improve. I believe my team can utilize this type of software process improvement and development in order to improve our process over all. If my team would look at where the flaws in our process lie and fix that flaw, I think our ability to develop a well working product.

As far as my future, I will be attempting to bring these new ways of development into my team, though because the way they currently develop is so entrenched, I feel it will go to deaf ears. What I will need to do is take this lesson into future jobs and eventually utilize it in my own managerial styles, when that time come. I believe it will suit me well in the future and am glad I learned it.

## Team Experience and Lessons Learned

Overall, Caliware learned about software development process, which is the set of tools, methods, and practices used to produce a software product. We learned and practiced in building a process with Scrum and XP. We also learned about building a mobile application for Android. Afterwards, we learned how to improve and assess our process by matching our process to using a standard, ISO 12207. We learned how to assess our process that is ISO 15504 compliant. In the assessment, we learned how to improve our process by finding weaknesses and improvement areas.

# Reference list

Humphrey, W. (1989). *Managing the software process.* Reading, Mass.: Addison-Wesley.

ISO/IEC 12207:2008, IEEE Std 12207-2008 : Systems and Software Engineering – Software Life Cycle Processes (2nd Edition 2008-02-01).

Jo, C. (n.d.-a). *Agile process & xp.* [PDF document]. Retrieved from CPSC 544 Advanced Software Process Web site:<https://moodle-2015-2016.fullerton.edu/course/view.php?id=8752>

Jo, C. (n.d.-b). *ISO/IEC 12207 IEEE Std 12207-2008 Systems and Software Engineering - Software Life Cycle Processes Introduction.* [PDF document]. Retrieved from CPSC 544 Advanced Software Process Web site:<https://moodle-2015-2016.fullerton.edu/course/view.php?id=8752>

Jo, C. (n.d.-c). *ISO/IEC 15504 Information Technology / Software Engineering – Process Assessment (SPICE) Introduction.* [PDF document]. Retrieved from CPSC 544 Advanced Software Process Web site: <https://moodle-2015-2016.fullerton.edu/course/view.php?id=8752>

Jo, C. (n.d.-d). *Advanced software process part I: Software process maturity 3. software process assessment.* [PDF document]. Retrieved from CPSC 544 Advanced Software Process Web site:<https://moodle-2015-2016.fullerton.edu/course/view.php?id=8752>

Jo, C. (n.d.-e). *Advanced software process part III: The defined process 14. the software engineering process group.* [PDF document]. Retrieved from CPSC 544 Advanced Software Process Web site:<https://moodle-2015-2016.fullerton.edu/course/view.php?id=8752>

Jo, C. (n.d.-f). *Scrum.* [PDF document]. Retrieved from CPSC 544 Advanced Software Process Web site:<https://moodle-2015-2016.fullerton.edu/course/view.php?id=8752>

Pikkarainen, M. (2006). *Mapping agile software development onto ISO 12207.* [PDF document]. Retrieved from CPSC 544 Advanced Software Process Web site:<https://moodle-2015-2016.fullerton.edu/course/view.php?id=8752>

Scrum alliance. (n.d.) *Learn about scrum.* Retrieved from<https://www.scrumalliance.org/why-scrum>

Wells, D. (1999). *The rules of extreme programming*. Retrieved from<http://www.extremeprogramming.org/rules.html>

# Appendix

# Team Charter

|  |  |  |
| --- | --- | --- |
| **Course Title** | CPSC 544 (50) Advanced Software Process | *All team members participated in the creation of this charter and agree with its content.* ***Date*** *08/31/2015* |
| **Instructor** | Dr. Chang-Hyun Jo |
| **Course Dates** | 08/24/2015 – 12/11/2015 |

**Team Members** (Contact Information)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Address (city, state, country) | Phone | Cell | Email |
| Timothy Cioffi-Dinkel | San Diego, CA | 951-775-6150 | 951-775-6150 | timothycioffi@csu.fullerton.edu |
| Anthony Farina | Bear, DE | 302-299-5495 | 302-299-5495 | afarina@csu.fullerton.edu |
| Joanna Hang | Fullerton, CA | 714-686-7401 | 714-686-7401 | johang@fullerton.edu |
| Lourdes Lopez | Turlock, CA | 209-205-0581 | 209-205-0581 | lourdeslopez@csu.fullerton.edu |
| David Sullivan | Fullerton, CA | 714-686-9488 | 714-686-9488 | dsullivan@fullerton.edu |

**Team Member Skill Inventory** (Areas individual members can contribute)

|  |  |
| --- | --- |
| Timothy Cioffi-Dinkel | * MySQL * Java/C++ * XML, HTML/CSS, PHP * Photoshop * Game Development, Unity/C# |
| Anthony Farina | * MySQL - Database administration and design * Java * Android Development * Microsoft Office * Google Web Toolkit |
| Joanna Hang | * SQL Server - Database administration and design * SQL Reporting Services * HTML, CSS, JavaScript, jQuery, Bootstrap, and Less * Microsoft Office * Adobe Photoshop, InDesign, Digital Publishing Suite |
| Lourdes Lopez | * Data Analytics - Complex queries using T-SQL or PL-SQL * Business Intelligence tools - Business Objects Web Intelligence, Business Objects Universe Design, Microsoft SSRS and Crystal Reports * Web Application Development with C# ASP.NET and JavaScript * Experience with agile development using Scrum methodologies * Microsoft Visio for documenting business process |
| David Sullivan | * iOS Development * Android Development * Java * C#, ASP.NET, JavaScript, Bootstrap * Microsoft Web API |

**Team Goals** (Project goals, team process goals, quality goals, etc.)

|  |
| --- |
| * Combine the unique skill sets of each of our team members, so that we can produce a successful project as well as learn from each other’s areas of expertise. * Learn about the software process by reviewing course material as well as experiencing the process as we work towards our course assignments. * Improve upon our existing knowledge of software development processes so that we will develop software in a more organized manner. * Keep track of our weekly assignments and goals so that we meet our deadlines and are well prepared for upcoming assignments. * Establish and maintain strong communication between teammates in order to facilitate success. |

**Team Roles** (Define roles of each member to achieve goals)

|  |  |
| --- | --- |
| Lourdes Lopez  Team Lead | * Act as the team liaison which includes communicating with the professor and/or class regarding the team’s progress and/or asking questions * Turn in assignments, take a screenshot of submitted assignment, and e-mail team members * Collaborate with Facilitator on agendas * Manage project timeline |
| Joanna Hang  Recorder | * Take meeting notes and put them in Google Drive * Document ideas and key decisions from meetings * Ensure team members know the purpose and goals for assignments |
| Timothy Cioffi-Dinkel  Facilitator | * Inform team of any changes to meetings * Assist team lead with agenda before meetings * Coordinate video calls/start video calls * Begin/end conference calls with a check-in with each team member * Ensure all team members participate and ask everyone for input. * Cover topics on the agenda * Manage meeting times |
| David Sullivan  System Architect | * Ensure the project is within the scope of technical requirements * Evaluate functional and non-functional requirements * Review project through a technical standpoint |
| Anthony Farina  Quality Assurance | * Ensure process meets users’ quality expectations * Extract collaborative assignments from Google Drive and convert documents to appropriate document type (Word, Excel, PDF, etc.) * Perform final review by checking format and reviewing assignments * Upload final copy to Google Drive and inform Team Lead that assignment is ready to turn in |

**Ground Rules**

|  |
| --- |
| * All team members will check Titanium daily. * All team members will check emails at least once per day and reply within 24 hours. * All team members will meet on Tuesday’s at 5:15 PM PST using Google Hangouts. * Google Hangouts will be the main tool for communication between team members. * Google Hangouts will be installed on each member’s mobile phone. * All team members will check Google Hangouts at least twice per day. * All team member will complete collaborative assignments using Google Drive. * All team members must be respectful, honest, and clear to all other team members at all times. * No plagiarism or cheating. Each team member will proof read each other’s work to prevent this. * Group will rotate team leaders each semester. * All team members will communicate and ask questions. * All team will equally contribute, show effort, and will not leave the work to a few individuals. * All team members will be supportive, willing to help and assist other team members. * Each member must be open to constructive criticism from each other team member. * Each member is responsible for their tasks. Tasks must be completed by the deadlines set by the professor and or the group. * If emergency arises that may affect deadlines, meetings, etc., team member will at least inform the team leader or if possible the entire group via Google Hangout about the situation so the group can plan accordingly. |

**Time Commitments/Availability** (Pacific Time)

|  |  |
| --- | --- |
| Timothy Cioffi-Dinkel | * M-F 5pm-8pm * Sat-Sun all day |
| Anthony Farina | * M-F 5pm-8pm * Sat-Sun all day |
| Joanna Hang | * M-F 5pm-8pm * Sat-Sun all day |
| Lourdes Lopez | * M-F 5pm - 8pm * Sat-Sun all day |
| David Sullivan | * M-F 5pm - 8pm * Sat-Sun all day |

**Conflict Management**

|  |
| --- |
| * In order to avoid conflict among team members, roles and responsibilities should be equally designated. * If conflicts arise, concerns should be submitted to the agenda and will be addressed to the whole team to settle conflicts. |

**Risk Management**

|  |
| --- |
| * Identify risk factors within team (meeting time conflicts , external commitments to work and family) and within the software process (training, organizational policies, configuration) * Prioritize tasks in order of importance and communicate to stakeholders * When problems arise, communicate with team and stakeholders * Actively monitor progress and reevaluate requirements |

**Team Evaluation Criteria**

|  |
| --- |
| * Evaluate each individual team member on their own work. * Evaluate each individual team member on their participation in group discussions. * Evaluate each individual team member on their ability to collaborate on the group project. * Evaluate each individual team member on their ability to produce material in a timely matter. * Evaluate each individual team member on their ability to fulfill role. |

# Team Evaluation

