Task & Deliverable View of Pedal

Figure 1: Pedal Software Process

Maintenance

**Workflows**

Requirements

Analysis

Design

Implementation

Test

Presentation

System Inception

Sprint Incepti

Elaboration

Construction

Transition

Sprint 3

Sprint Incept

Elaboration

Construction

Transition

Sprint 2

Sprint Inception

Elaboration

Construction

Transition

Sprint 1

**Phases**

System Completion

**System Inception**

**Purpose**

The purpose of the System Inception phase is to gain a high-level (big picture) understanding of system.

**Discussion**

This phase should be done quickly. The goals are to determine the major actors, to determine the major functions the customer requires the system to perform, to identify major risks and candidate technologies for use, and to determine a very high level tentative architecture (or alternatives to be explored).

**Tasks**

|  |  |  |
| --- | --- | --- |
| **Workflow** | **Task** | **Deliverable** |
| Requirements | Identify the major actors and use cases of the system. | Initial Use Case Model |
| Analysis | * Analyze initial risks and technology needs. * Estimate the effort & calendar time required to build the system. | * Risk Analysis * Technology Plan * Cost Analysis |
| Design | Identify possible architectures. | Deployment Model |
| Implementation | None. | None |
| Test | Walkthrough with customer. | Minutes of meeting |
| Presentation | * Determine who, what & when for Sprint 1 Inception. * Create a system inception presentation. | Presentation |

**Sprints**

***Sprint Inception***

**Purpose**

The purpose of the sprint inception phase is to determine the system functionality that will be implemented during the sprint.

**Discussion**

This phase varies by sprint. The focus in sprint 1 is to produce an increment of the system that implements a system with the minimal required functionality. Additional, sprints should incrementally produce systems that provide additional functionality. In each case, sprint inception concentrates on determining and understanding the functionality, which is focus of the sprint.

**Tasks**

|  |  |  |
| --- | --- | --- |
| **Workflow** | **Task** | **Deliverable** |
| Requirements | Identify the functionality addressed in the sprint. | Sprint Use Case Model |
| Analysis | * Update the risk analysis and technology plan focusing on the functionality addressed in the sprint. * Estimate the effort & calendar time required to build the sprint. | * Sprint Risk Analysis * Detailed Tech Training Plan * Cost Analysis |
| Design | Update the deployment model focusing on the functionality addressed in the sprint. | Sprint Deployment Model |
| Implementation | Begin technology training. | None |
| Test | Walkthrough of deliverables. | Minutes of meeting |
| Presentation | * Determine who, what & when for elaboration. * Create a sprint inception presentation. | Presentation |

***Elaboration***

**Purpose**

The purpose of the elaboration phase is to create an architectural level design for the sprint.

**Discussion**

The focus of the elaboration phase is on exploring and refining the system architecture. Producing system code is not an objective of the phase. However, defining the architecture will produce questions that are best addressed by writing code and doing so is not only acceptable but is necessary. As a word of caution, coding on the system during Elaboration is premature. A detailed design has not been addressed and coding by multiple people without addressing a detailed design is a recipe for disaster in a large project.

**Tasks**

|  |  |  |
| --- | --- | --- |
| **Workflow** | **Task** | **Deliverable** |
| Requirements | Review the use case model for the sprint and update as necessary. | Updated Sprint Use Case Model |
| Analysis | Perform an object-oriented analysis of the sprint.   * Describe each use case using the analysis level objects identified. * Focus on the essential operation of the system. | Sequence or Communication Diagrams for each use case |
| Design | * Identify subsystems and subsystem interfaces. * Refine the deployment diagram for the sprint taking in account the architectural implications of software frameworks that are to be used. * Map the components identified on to the architecture and document in the deployment diagram. * Analyze the relationships between the classes derived from the analysis. * Determine which requirements must be fulfilled by the sprints deliverable. | * Component Diagram * Updated Deployment Model * Class Diagram * Acceptance Plan |
| Implementation | Technology training focused on the technologies necessary to implement the system. At the maximum rough, high-level sketches of selected components might be produced as a learning activity. (They must be viewed as throw away.) | None |
| Test | Walkthroughs of the use case model and the architectural design. | Minutes of the meetings |
| Presentation | * Determine who, what & when for construction. * Create an elaboration presentation. | Presentation |

***Construction***

**Purpose**

The purpose of the construction phase is to create a system implementing the functionality specified for the sprint.

**Discussion**

The major workflows for construction involve creating a detailed design during the design workflow, implementing the design during the implementation workflow, and performing unit and integration testing in the test workflow.

**Tasks**

|  |  |  |
| --- | --- | --- |
| **Workflow** | **Task** | **Deliverable** |
| Requirements | Review the use case model for the sprint and update as necessary. | Updated Sprint Use Case Model |
| Analysis | Review the analysis classes and update if necessary. | Updated Sequence or Communication Diagrams for each use case |
| Design | * Determine design level classes for the sprint taking into account any software frameworks employed. * Map the design classes onto the component model and determine the implementation artifacts on which each component maps. * Using this information update the component model and the deployment model. * Identify the data entities, the attributes of each entity, and the primary & foreign keys for the sprint (i.e. design the database). * Design screen templates. Design the transitions between major screens. * If the system is a real time system or if it is helpful, analyze the behavior of the system in terms of states, transitions between the states, and transition triggers. * For Sprints 2 & 3, describe how the design limits duplication of functionality & integrates new features with the existing features. * Break the coding for the sprint into units. * Determine Acceptance Test procedures * Determine how units will be tested. * Determine how units will be integrated. | * Design Class Diagrams organized by component * Updated Component Model * Updated Deployment model * ERD for the sprint * Screen templates, Screen Navigation * (optionally) State Transition Diagram * Acceptance Procedures * Unit test plan & test procedures * Integration test plan & test procedures |
| Implementation | Code each unit and then integrate the units into the application. | Source Code & binary executables. |
| Test | * Perform unit testing as each unit is completed. * Perform integration testing as each unit is integrated into the application * Analyze the quality of software using departmental coding standards. | * Unit test report * Integration test report * Software quality report |
| Presentation | * Freeze the code and produce a list of currently known issues. * Determine who, what & when for transition. * Create construction presentation. | Presentation |

***Transition***

**Purpose**

The purpose of the transition phase is, as the name implies, to transition from construction of the product through product delivery to product use.

**Discussion**

In the transition phase the product is delivered and the delivered product is evaluated. In Sprint 1 the focus of the evaluation is the selected architecture and the implementation technologies used. If either is found lacking, Sprint 2 may be used to redo Sprint 1. In Sprint 2 the focus of the evaluation is the functionality of the current increment of the product, the quality of the current design (both architectural and detailed), and the quality of the current implementation. In Sprint 3 the focus of the evaluation is preparation for maintenance.

**Tasks**

|  |  |  |
| --- | --- | --- |
| **Workflow** | **Task** | **Deliverable** |
| Requirements | Review and update the use case model. It is important that insights into the requirements that were gained during the sprint be captured. | Updated Use Case Model. |
| Analysis | Evaluate the decisions made in:   * defining the architecture, * evaluate if the architecture supports or inhibits future expansion, * selecting implementation technologies * creating the detailed design. Reassess system risks. | * Updated Risk Analysis * Updated Technology Plan |
| Design | Update the architectural and detailed designs based upon the results of the analysis workflow. | * Updated Component diagram * Deployment, diagram * Sequence diagram and * Class diagrams |
| Implementation | * Source code for the product is frozen at the end of construction and cannot be changed during transition. * Defects should be recorded as issues, prioritized, and corrected in later sprints or maintenance. * Modify configuration files to transition the product to a production state. | * Configuration files |
| Test | Perform the acceptance test. | Acceptance test report |
| Presentation | * Determine who, what, & where for sprint inception for the next sprint. In the case of the last sprint determine who, what, & where for maintenance. * Create a delivery presentation | Presentation |

**Maintenance**

**Purpose**

The purpose of the maintenance phase is to address post-delivery issues and defects in the product.

**Discussion**

There are three primary sources of changes to a released product that must be evaluated. Even the most carefully designed and implemented product will contain defects that require the released product be changed. In the case of new development it is likely that there is a list of remaining issues of low enough priority that they were not corrected in the delivery release. The customer will also request changes in the released product. These requests should be evaluated, prioritized, and added to the issues list. Changes in the operating environment (i.e. new laws and changed business procedures, for example) or changes in the available technology require changes to the delivery release be made. Maintenance is the process of making the needed changes.

Maintenance works off a prioritized issues list. Each iteration of the maintenance phase should lead to a separate update release. The process is as follows:

Take the highest priority issue, find all closely related issues in the list, and determine a subset of issues to address in the next release. For the subset of issues perform maintenance tasks as outlined below.

As new issues are identified, they should be prioritized and added to the issues list rather than addressed as they arise. Focus should be kept on the selected issues for the current update release.

It is worth noting that some issues, such as security defects or issues that prohibit the use of the system are so critical that they take precedence and should be addressed immediately. When issues of this magnitude arise the team must make decisions in collaboration with the customer and upper management (professor). In these cases, the current release work should be terminated, code should be rolled back if necessary, and a maintenance update release should be executed for the critical issue as outlined below.

**Tasks**

(see next page)

For each release:

|  |  |  |
| --- | --- | --- |
| **Workflow** | **Task** | **Deliverable** |
| Requirements | * Determine the issue set to address in this release. * Review the requirements of the system related to the issue. * If the issue introduces a new requirement, modify the requirement model. | Modified Use Case Model |
| Analysis | Analyze the issues for the release in terms of design and source code. The goals of the analysis are to:   * determine the cause of the issue * determine changes to the design * determine changes to the code * determine an overall regression test plan or exist the existing plan * determine the time and effort required to address the issues in this release. | * Unit test plan Integration test plan * (Updated) Regression test plan * Cost analysis |
| Design | Based upon the analysis make the required changes to the design. | * Modified design * Unit test procedures * Integration test procedures |
| Implementation | Modify the source code to implement the changes. | * Source code * Executables |
| Test | * Preform unit testing * Perform integration testing * Perform overall regression testing. | * Unit test report * Integration test report * Regression test report |
| Presentation | * Freeze the code for this release. * Remove resolved issues from the issue list. * Release the update. * Create a release presentation. | * Presentation * Updated issues list |

**System Completion**

**Purpose**

The purpose of the system completion phase is to improve your software development skills by reflecting upon the completed project.

**Discussion**

Most software development projects have a wrap-up or debriefing phase where the development team concentrates on the process used to complete the project. The process is to analyze what worked and what did not and to create a list of lessons learned. It requires an honest appraisal of each person’s work in terms of both individual and peer evaluation. The goal is simply to learn to build better systems in a better way. In order to elicit open and honest discussion the results cannot be used for evaluation of participants.

In some organizations this phase may include a formal process using metrics collected during the project as feedback to improve the software process for the organization as a whole. In these organizations this phase is a part of a continual process improvement program.

**Tasks**

|  |  |
| --- | --- |
| **Task** | **Deliverable** |
| * Determine what refactoring needs to be done in order to improve the code for future maintenance and development. * Identify the maintenance changes you made that had unanticipated consequences. * Identify the lessons learned during the project by both the team and by each team member. * Create a final project presentation that summarizes the results of the above tasks. | * List of lessons learned. * Final Project Presentation |