Software Engineering Lifecycle Reading

# Requirements Analysis

Inputs

* Understanding the requirements of the system to be built through cutsomer interviews or market analysis.
* Misunderstood requirements can be expensive to fix later in the project.

Outputs:

* Requirements definition documents (RDD) or requirements specification document.
* In structured analysis, for example, the resulting document includes dataflow and control flow diagrams, data dictionary, and entity relationships diagram. Object-oriented analysis leads to use case modeling, class modeling, and dynamic modeling diagrams and descriptions

Principles:

* Requirements definition document is intended for communication with the customers and it must be in a language understandable to those unfamilar with technical aspects of the software construction.
* At the same time, it should capture all requirements essential for successful completion of the project. Typically it involves semi-formal diagrams and descriptions in a natural language like English.

Validation Activity:

* Requirements definition document is validated through multiple organized meetings with clients to check that the documented requirements meet their expectations
* Rapid prototyping is one technique for validating requirements and seeking client feedback.
* This is a multi-step process, wherein this step is repeated several times to ensure validity
* There is also an internal validation activity to check that the document is consistent and complete.
  + An inconsistent document contains conflicting requirements that cannot be simultaneously fulfilled.
  + An inconsistent document just cannot be implemented. Completeness refers to documenting all essential requirements.
  + The document should be relatively complete with respect to implicit requirements.

# Design & Specifications

Inputs:

* Requirements definition document

Outputs:

* Design and specification document.
* It includes module design diagrams, object coupling diagrams, and interface descriptions of each module.
* Unlike the requirements definition document, intended to be understandable by non-software specialists such as customers and managers, module specifications are intended for software developers.
* They can be described in a formal or mathematical language suitable for software professionals.

Principles:

* The central objective of the design stage is to modularize the system to be developed in such a way that they can developed relatively independently and integrated easily.
* In module design, the key principles are cohesion and coupling. Cohesion is a measure of how "single minded" a module is. Each module should have a high level of functional cohesion. Coincidental cohesion, for example, is undesirable.
* Coupling is a measure of how inter-related the modules are. Parameteric coupling is desirable whereas global (or coomon) coupling is undesirable.
* The objective of design is to minimize coupling and maximize cohesion. In component-based software engineering, the design stage aims to reuse pre-existing components.

Validation Activity:

* The specification document can be validated by a design review committee by comparing it with the requirements definition document.
* Since requirements definitions are not in a formal language, mechanical comparison is not possible. However, it is possible to use mechanical means to check internal consistency and (relative) completeness of specifications

# Implementation

Inputs:

* Design & specification document

Outputs:

* Code, code documentation and performance documentation

Principles:

* Each module of a working system needs to be implemented (unless one previously exists and can be reused). Implementation of a module involves producing code in a programming language to meet the specification of that module.
* Ideally, the code should be reusable. Any kind of implementation technique can be used, as long as it satisfies the dual objectives of producing correct and efficient code. Of course, the code should also be simple and understandable.
* In general, there are alternative implementations to meet the same specification. There is usually no best one. Important space-time trade-offs need to be made.

Verification & Validation Activities:

* Quality Assurance Stage

# Quality Assurance

Inputs

* All documents including requirements definition document, design & specification document, performance documentation, and documented code

Outputs

* Reports on inconsistencies across stages and within stages

Principles:

* Unit and system-wide checking

Verification & Validation Activities:

* There are alternative methods to check if a module implementation meets its specification, including formal verification, code inspection/tracing, model checking, and unit testing.
* These activities will be discussed in detail in this course with emphasis on verification and testing. The validation activity is integrated testing to check if the system satisfies the ultimate customer requirements.

# Support or Maintenance Phase

Input:

* All documents including requirements definition document, design & specification document, performance documentation, and documented code

Outputs:

* Modified artifacts

Principles:

* Maintenence may be needed for enhancement of functionality and performance. Or it may be corrective (to fix errors) or adaptive (to make the system work in a new environment).

Verification & Validation Activities:

* It is essential to verify & validate that modifications produce desired effects as well as check that nothing else was compromised due to the changes. The principle of regression testing is to minimize re-testing that is essential when code is modified to fix an error or improve functionality