Software Engineering

Lecture 6:

Software Design

SRS: Cont. Lecture 5

- ***The basic issues that the SRS shall address are the following:***

1. ***Functionality****: What is the software supposed to do?*
2. ***External interface****: How dose the software interact with people, the system's hardware, other hardware, and other software?*
3. ***Performance****. What is the speed, availability, response time, recovery time of various software function.*
4. ***Attributes:*** *What are the portability, correctness, maintainability, security, etc. considerations?*
5. ***design constraints imposed on an implementation****: Are there ant required stander in effect, implementation language, policies for database integrity, resource limits, operating environment etc.*

SRS: Cont. Lecture 5

- ***Remember that there is no "Perfect SRS". However, SRS should be:***

1. ***Correct****: each requirement represents something required by the target system.*
2. ***Unambiguous****: every requirement in SRS has only one interpretation*
3. ***Complete****: everything the target system should do is included in SRS (no sections are marked TBD-to be determined).*
4. ***Verifiable****: there exists some finite process with which a person/machine can check that the actual as-built software product meets the requirements.*
5. ***Consistent*** *in behavior and terms.*

SRS: Cont. Lecture 5

6. ***Understandable*** *by customers.*
7. ***Modifiable****: changes can be made easily, completely and consistently.*
8. ***Design independent****: doesn't imply specific software architecture or algorithm.*
9. ***Concise****: shorter is better.*
10. ***Organized****: requirements in SRS are easy to locate; related requirements are together.*
11. ***Traceable****: each requirement is able to be referenced for later use (by the using paragraph numbers, one requirement in each paragraph, or by using convention for indication requirements)*

Software Design

- *Software design is a process to transform **user requirements** into some suitable form and translated into a "blueprint", which helps the programmer in software **coding and implementation**.*
- ***During the design process the SRS are transformed into design models that describe the details of the data structures, system architecture, interface, and components. At the end of the design process a design specification document is produced. This document is composed of the design models that describe the data, architecture, interfaces and components.***
- *Software design is the first step in **Software Design Life Cycle**, which moves the concentration from problem domain to solution domain. It tries to specify how to fulfill the requirements mentioned in SRS.*

Software Design

- *Design is where customer requirements, business needs, and technical considerations all come together in the formulation of a product or system.*
- *The design model can be assessed and tests for quality and be improved before code is generated.*

1. ***Does the design contain errors, inconsistencies, or omissions?***
2. ***Are there better design alternatives?***
3. ***Can the design be implemented within the constraints, schedule, and cost that have been established?***

Software Design Process

- ***Software design*** *is an iterative process through which requirements are translated into a blueprint for constructing the software.*
  1. *Design begins at a* ***high level of abstraction*** *that can be directly traced back to the data, functional, and behavioural requirements.*
  2. *As design iteration occurs, subsequent refinement leads to design representations at much* ***lower levels of abstraction.***
- *The main goal of the process design is to produce a model or representation that will later be built.*

Software Design Process

Each element of the SRS document provides information that is necessary to create the 6 design models:

- ***The data design.***
- ***The architectural design.***
- ***The interface design.***
- ***The component-level.***
- ***Database Design.***
- ***Deployment-level***

Software Design Process

1. ***Data design:***
   - *Creates a model of data and objects that is represented at a high level of abstraction.*
   - *Part of the data design may occur in conjunction with the design of software architecture. More detailed data design occurs as each software component is designed.*
2. ***Architectural design:***
   - *Identify the overall structure of the system, the principal components (sub-systems, modules), their relationships and how they are distributed.*
   - *Depicts the overall layout of the software.*

Software Design Process

3. ***Interface design:***
   - *Tells how information flows into and out of the system and how it is communicated among the components defined as part of the architecture.*
   - *Includes the user interface, external interfaces, and internal interfaces.*
4. ***Component-level design elements:***
   - *Describes the internal detail of each software component by way of data structure definitions, algorithms, and interface specifications.*
   - *Take each system component and design how it will operate.*

Software Design Process

5. ***Database design***
   - *Design the system data structures and how these are to be represented in a database*
   - *Work depends on whether existing database is to be reused or a new database is to be created*
6. ***Deployment-level design elements:***
   - *Indicates how software functionality and subsystems will be allocated within the physical computing environment that will support the software.*

- *Each design product is reviewed for quality before moving to the next phase of software development.*
- *At the end of the design process a design model and* ***specification document*** *is produced.*

SD: Design Verification

- *The output of software design process is* ***design documentation***, ***pseudo codes***, ***detailed logic diagrams***, ***process diagrams***, *and* ***detailed description of all functional or non-functional requirements***. *The next phase, which is the implementation of software, depends on all outputs mentioned above.*
- *It is then becomes necessary to verify the output before proceeding to the next phase. The early any mistake is detected, the better it is or it might not be detected until testing of the product.*
- *By structured verification and validation, reviewers can detect defects that might be caused by overlooking some conditions. A good design review is important for good software design, accuracy and quality.*

Software Design

- ***Characteristics of a Good Design***

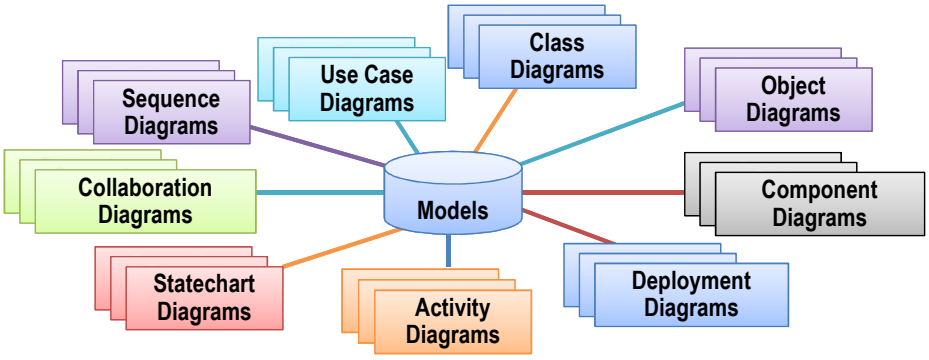
1. *A design should be modular; that is, the software should be logically partitioned into elements that perform specific functions and subfunctions.*
2. *The design must be readable, understandable guide for those who generate code and for those who test and support the software.*
3. *The design should provide a complete picture of the software, addressing the data, functional and behavioral domains from an implementation perspective.*

SD: Unified Modelling Language

- *System modeling is the process of developing abstract models of a system, with each model presenting a different view or perspective of that system.*
- *System modeling has now come to mean representing a system using some kind of graphical notation, which is now almost always based on notations in the* ***Unified Modeling Language (UML).***
- *System modelling helps the analyst to understand the functionality of the system and models are used to communicate with customers.*

SD: Unified Modelling Language

- ***UML:*** *Devised by the developers of object-oriented analysis and design methods, it has become an effective standard for software modelling and it as nine different notations*



SD: Unified Modelling Language

- *The UML has many diagram types and so supports the creation of many different types of system model.*

1. ***Activity diagrams****, which show the activities involved in a process or in data processing.*
2. ***Use case diagrams****, which show the interactions between a system and its environment.*
3. ***Sequence diagrams****, which show interactions between actors and the system and between system components.*
4. ***Class diagrams****, which show the object classes in the system and the associations between these classes.*
5. ***State diagrams****, which show how the system reacts to internal and external events.*