Created & Compiled
By Saroj Dahal

Chapter 1 - Software project management concepts.

Waterfall model (linear sequential model)

1. Communication (Requirement Analysis)
2. Planning & System Analysis
3. System Design & Modelings
4. Implementation (Development)
5. Testing/Debugging
6. Deployment
7. Update / Maintenance.

Prototyping Model

Customer sets general objective for software but doesn't properly identify detailed system behavior. Prototyping models tries to capture requirements of customer in detail through a series of quick design and evolution.

This model beings with requirement gathering,then a quick design occurs which leads to develop a prototype. Customer evaluates the prototype and uses it to refine requirements. Then iteration occurs as a prototype is modified to satisfy customers need.

Spiral Model

(also known as Risk driven model; proposed by Bohem).

4 P's of Project Management : People, Process, Product .. Project

W5HH principle for Project planning

Created & Compiled
By Saroj Dahal

Chapter 2 - Software Metric

You cannot control what you cannot measure.

Unlike other subjects, software metrics are not exactly factual as compared to physics and science, but from the metric we can estimate and understand a lot of things.

Metrics :

- Product Metric : size, complexity, features (funacality), performance, reliability, quality level
- Process Metric : Effort required, Time required, Failure rate, productivity.
- Project Metric : Cost, Schedule, No of staff (developer), changes in project.
-

Measure, Metric, Indicator :
Measure : Quantitative amount .
Metric : That measure which possesses a given attribute.
Indicator : Metric or combination of metric that gives insights on software…

Chapter 3 - Risks

Steps for Risk Mitigation:

1. Identify the risk.
2. Perform risk assessment.
3. Prioritize risk.
4. Track risks.
5. Implement and monitor progress.

Risk Exposure (RE) = Prob. x Cost = PC

Created & Compiled
By Saroj Dahal

Chapter 4 - Software Quality Assurance (SQA)

Quality Control : It is the series of inspection, reviews and tests done throughout the software development process in order to ensure that the requirements are met along with the quality.

Quality Control contains the feedback loop, through which the constant changes for feedback is taken and is done.
The main purpose of quality control is to assure that the quality software is produced and all requirements are met.
Process of doing quality control:

1. Fully automated
2. Fully manual
3. Combined (Automated tools with human interactions).

Cost of Quality : All the costs/expenses done in order to achieve the quality in software and to ensure quality related thing is Cost of Quality.
Types :

1. Prevention Cost = Quality planning, FTR, Training, Testing equipment.
2. Appraisal Cost. = Process introspection , maintenance costing.
3. Failure Cost. = failure, repair work etc.

ISO Standards :
International Standard Organization. Has Quality standards starting from 9001 , 9002 and 9003. All for the quality.

9001 : Concerns with proper documentation of the company which makes the product.
9002 : Concerns with quality for the company which does manufacturing other than making design products.
9003 : Concerns with quality for the company which does installation and testing things.

Created & Compiled
By Saroj Dahal

Chapter 5 : Software Configuration Management (SCM)

Software Configuration management is the art of identifying, organizing and controlling modifications to the software being built by a programming team.
The goal is to maximize productivity by minimizing mistakes.

SCM Purpose :

1. To manage changes efficiently.
2. To facilitate construction of different versions of applications.
3. To ensure Software quality is maintained as the configuration evolves over the time.

Software Configuration Item (SCI) : A configuration item (CI) is any service component, infrastructure element, or other item that needs to be managed in order to ensure the successful delivery of services.

A SCI could be considered to be a single section of a large specification or one test case in a large suite of tests.

BaseLine

A baseline is a milestone in the development of software that is marked by the delivery of one or more software configuration items and

SCM Tasks :

1. Identifying (Tracking).
2. Version Control.
3. Change Control
4. Configuration auditing.
5. Reporting.

VCS : git, subversion, mercurial etc.

VCS initial ways :
-> lock-modify-unlock
-> copy-modify-merge

Chapter 6 : Analysis concept and principles

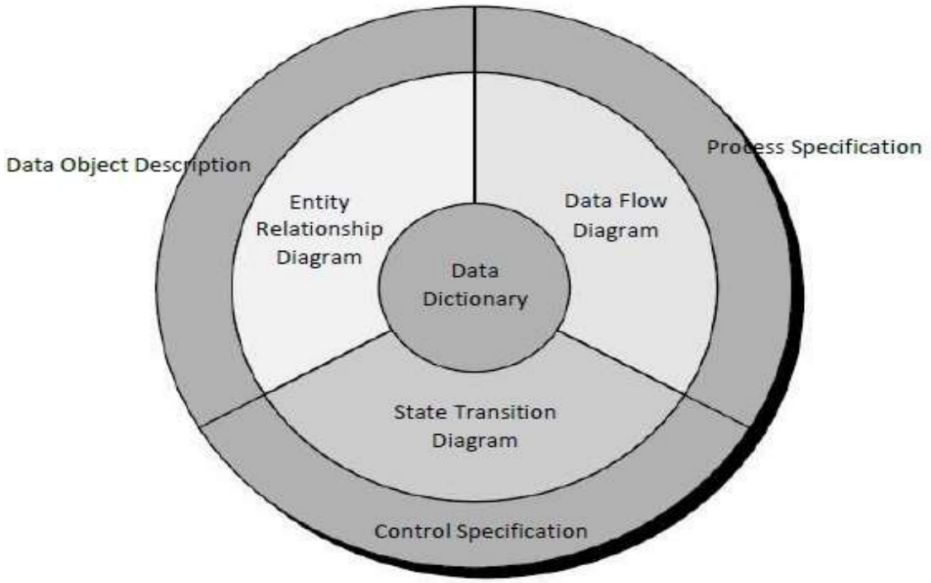
Requirement elicitation = Requirement Gathering.

Software Requirement Specifications (SRS):
The specification and details which are required before creating a work product for the software is SRS.
SRS Document usually contains three things:

1. A purpose
2. An overall description and its context.
3. Specific requirements.

Objectives of Analysis Model :

1. Must describe what the customer requires.
2. Must establish a basis for the development of software design.
3. Should be validated also, after software development.



The structure of the analysis model

Cardinality and Modality

The cardinality describes the number of relations an object possesses for another object. On the other hand, modality describes whether the relation is required to be made or not.
Example of cardinality : 1:1 , 1:n (1 to many), m:n (many to many)

Data Dictionary :

It is the set of all the list of data elements of the software with rigid definition such that the end user and system analyst can understand all the process of softwares from input to output, processing and even the store process.
The data dictionaries format varies from tool to tool ; but the common format usually includes the following information :

1. Name
2. Alias
3. where-use/how-used
4. Content description
5. Supplementary information.

Chapter 7: Design

10 Principles Of Software Design :

Should not suffer from “Tunnel Vision” –
While designing the process, it should not suffer from “tunnel vision” which means that it should not only focus on completing or achieving the aim but on other effects also.

Traceable to analysis model –
The design process should be traceable to the analysis model which means it should satisfy all the requirements that software requires to develop a high-quality product.

Should not “Reinvent The Wheel” –
The design process should not reinvent the wheel, that means it should not waste time or effort in creating things that already exist. Due to this, the overall development will increase.

Minimize Intellectual distance –
The design process should reduce the gap between real-world problems and software solutions for that problem meaning it should simply minimize intellectual distance.

Exhibit uniformity and integration –
The design should display uniformity which means it should be uniform throughout the process without any change. Integration means it should mix or combine all parts of software i.e. subsystems into one system.

Accommodate change –
The software should be designed in such a way that it accommodates the change implying that the software should adjust to the change that is required to be done as per the user’s need.

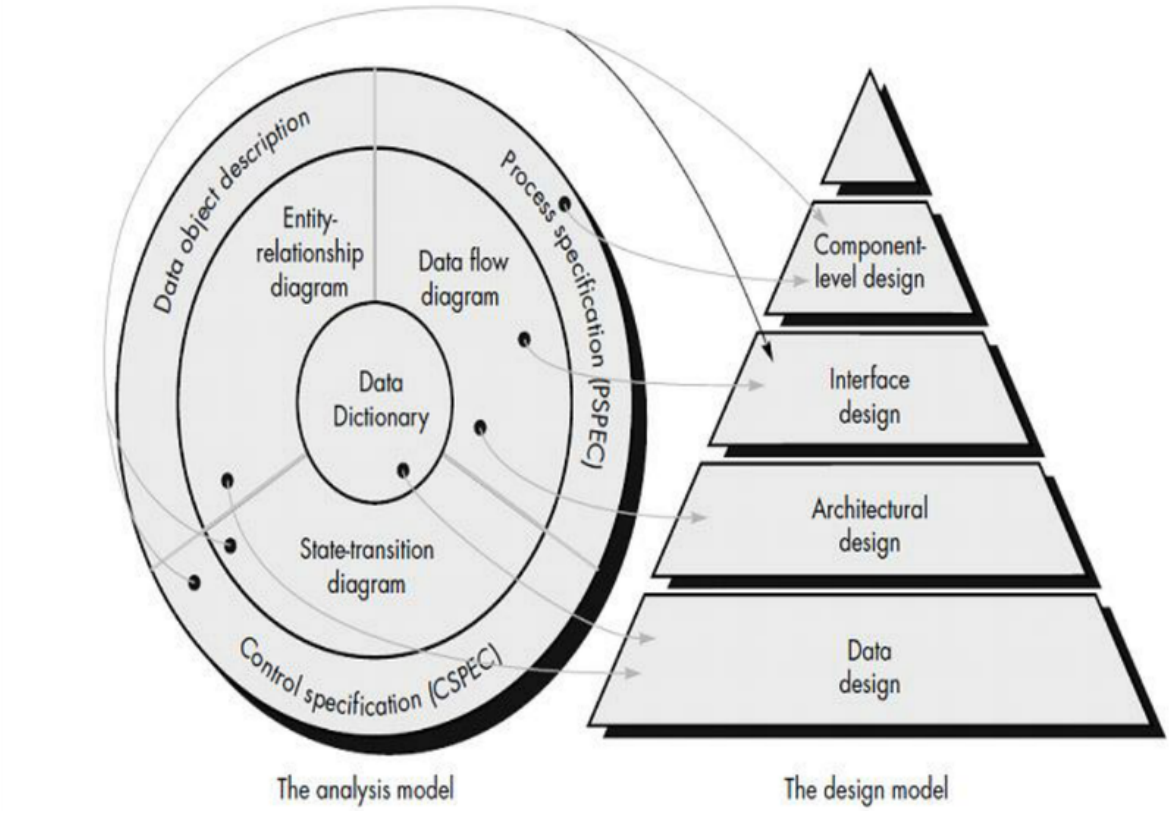
Degrade gently –

The software should be designed in such a way that it degrades gracefully which means it should work properly even if an error occurs during the execution.

Assessed or quality –
The design should be assessed or evaluated for the quality meaning that during the evaluation, the quality of the design needs to be checked and focused on.

Review to discover errors –
The design should be reviewed which means that the overall evaluation should be done to check if there is any error present or if it can be minimized.

Design is not coding and coding is not design –
Design means describing the logic of the program to solve any problem and coding is a type of language that is used for the implementation of a design.



Created & Compiled
By Saroj Dahal

Design Architecture (Architectural Design)

1. Data Centered Architecture. (Data is stored in the center; and accessed by others).
2. Data Flow Architecture : Used when input data have to be transformed into output data through a series of components and with computations. (pipes and filters in fig.)
3. Call and return Architecture : Used when the program/software needs to scale and modify at times.
4. Object oriented architecture : used when a system needs to encapsulate its information; and where the components communicate with each other from message passing.
5. Layered Architecture : A number of different layers are defined with each layer performing a well- defined set of operations. Each layer will do some operations that become closer to the machine instruction set progressively.

⚫ At the outer layer, components will receive the user interface operations and at the inner layers, components will perform the operating system interfacing(communication and coordination with OS)

⚫ Intermediate layers to utility services and application software functions.

What is a Test case?

A test case is a document, which has a set of test data, preconditions, expected results and postconditions, developed for a particular test scenario in order to verify compliance against a specific requirement.

Test Case acts as the starting point for the test execution, and after applying a set of input values, the application has a definitive outcome and leaves the system at some end point or also known as execution postcondition.

Typical Test Case Parameters:

- Test Case ID
- Test Scenario
- Test Case Description
- Test Steps
- Prerequisite
- Test Data
- Expected Result
- Test Parameters
- Actual Result
- Environment Information
- Comments

Types of Black Box Testing.

1. Graph-based Testing
2. Equivalence Partitioning
3. Boundary Value Analysis (BVA)
4. Comparison Testing
5. Orthogonal Array Testing.

Encapsulation :

Encapsulation is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as data hiding.

OOAD:

Object Oriented Analysis and Design.

Two parts :

OOA (Object oriented Analysis) + OOD (Object Oriented Design)

OOA (Object Oriented Analysis):

-> Identifying Objects and its entities & methods; while communicating with customers.

-> Finding and describing objects in the problem domain.

-> Investigation of problems and requirements.

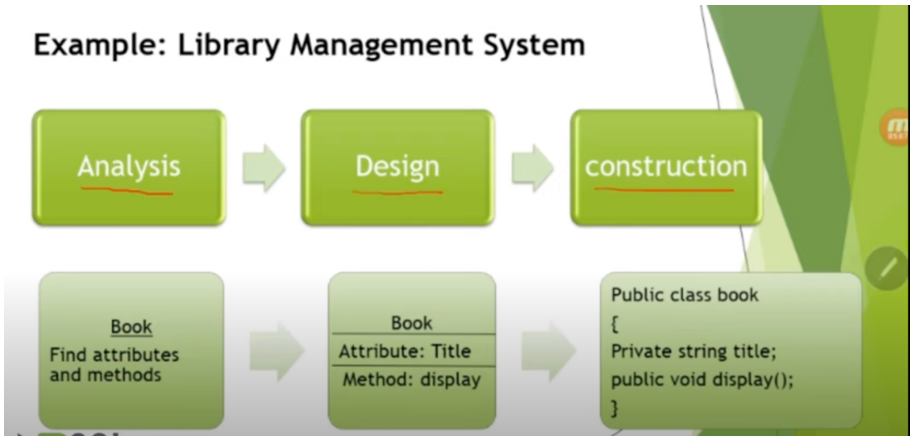
-> Requirement analysis.

-> Object based analysis.

Ex : CRC cards, UML

OOD (Object Oriented Design):
-> Defining Objects and their collaboration to fulfill the requirements.
-> Conceptual solution . Less detailed.
-> System Modeling.

Ex : Usecase Diagram (Diagram having actors).



Domain Analysis

Domain Analysis is the process of analyzing related software systems in a domain to find their common and variable parts. It is a key method for realizing systematic software reuse.

This activity, called domain analysis, is performed when an organization wants to create a library of reusable classes (components) that will be broadly applicable to an entire category of applications.

Created & Compiled
By Saroj Dahal

Design Patterns:

Design Patterns are typical solutions to commonly occurring problems in software design.

1. Creational Design Patterns (Singleton, factory, builder)
2. Behavioral Design patterns (State, Observer, iterators)
3. Structural Design Patterns (Decorators, Adaptors)