Theory of Practical Correspondence

A **Software development methodology** is division of software development work into various stages containing activities for better planning and management. It is a well-defined, structured sequence of stages in software engineering to develop the intended software product.

Types of Software developing life cycles (SDLC)

- Waterfall Model
- Spiral Model
- Prototype Model
- Iterative Model

Waterfall Model

Waterfall Model also known as the 'linear-sequential' life cycle model is one of the classic life-cycle models. In this model each phase must be completed before moving onto the next one. Review process is assigned at the end of each phase to check that the project is on the right track.

Advantages of the waterfall model

- Simplest to understand and use.
- Every phase is independent of other phases and is processed and completed separately.
- Usually used for smaller projects and for projects where the requirements are clearly outlined.

Disadvantages of the waterfall model

1. No working software is produced till late in the life cycle.
2. High level of uncertainty and risks.
3. It is not a good choice for big or on-going projects.

Iterative Model

The Iterative model is also known as multi-waterfall cycle. Cycles are divided into smaller and easily manageable iterations. Each iteration passes through a number of sequential phases, so after each cycle we get working software.

Advantages of the iterative model

1. Produces working software early during the lifecycle.
2. More flexible as requirement changes can be implemented at low cost.
3. As the iterations are small, testing and debugging is easier.
4. As the risks can be identified and resolved during each iteration, there is Low risks factors.

Disadvantages of the iterative model

1. This model comprises of phases that are rigid and do not overlap.
2. As not all the requirements are gathered before starting the development; this could lead to problems related to system architecture at later iterations.

Spiral Model

The spiral model is similar to the iterative model, but concentrates more on risk analysis. System requirements are defined in as much detail as possible by involving various users and various aspects of the system.

The most crucial step in the spiral model is creation of a preliminary design of the system as it helps in developing cost-effective strategies for working on a project.

Using the preliminary design, a prototype is developed. This is a scaled down system, which represents an approximate characteristics of the final output.

Consecutive prototypes are evolved through the following procedure

- Weaknesses, Strengths and risks of the previous prototype are evaluated
- Requirements for the new prototype are decided.
- Design and Planning of the new prototype begins
- Developing and testing the new prototype are carried out.

Advantages of the spiral model

1. Good for critical and large projects.
2. Working software is produced early during the lifecycle.
3. Large amount of risk analysis.

Disadvantages of the spiral model

1. Involves higher cost.
2. Not suitable for smaller projects.

3. Project success depends on the risk analysis phase - hence, it requires highly specific expertise in risk analysis.

Prototype Model

The prototype model is used to rectify the limitations of waterfall model. In this model, instead of fixing the requirements before coding or design, a prototype is built to understand the requirements. This prototype is based on the current requirements. Examining this prototype, the client gets a better understanding of the features of the final product. The processes involved in the prototyping approach are:

Advantages of the prototype model

1. Benefits from user input.
2. As a working model of the system is provided, users get a better understanding of the system that is being developed.
3. Errors and risks can be detected at a much earlier stage, as the system is developed using prototypes.

Disadvantages of the prototype

1. Increases complexity of the overall system.
2. Involves exploratory methodology and therefore involves higher risk.
3. Involves implementing and then repairing the way a system is built, so errors are an inherent part of the development process.

Our project is based on Iterative Waterfall Model as we made our project layerwise, means it passed through a sequence of activities at each phase. At each phase all the activities right from requirement phase till testing, were performed giving us our appropriate reslut.This was laborious but then it was the need of this project to go through each and every step in each phase. The other models like prototype and spiral won’t have allowed to do this so we chose Iterative Waterfall Model.

Software Requirements Specification Document

A software requirements specification document typically contains:

A full description of the software’s functionality and purpose.

Details about the software’s speed, response time, availability, portability, maintainability, recovery speed and more.

Description of how to use the software using use cases.

The definition of the applications interaction with hardware and program

Non-functional requirements (performance engineering requirements, quality standards, or design constraints).

Importance

A SRS clears the developer's goals of the software and on what they should focus on.

It allows them to:

Save time on communication

Minimize development efforts

Gives the customer feedback

Eliminate task duplication

Facilitate the transfer to new users or to new machines

Breaks problems down into parts

Serves as the main document to verify the validation and testing processes

Referring to past SRS documents helps identify deficiencies and process flaws.

There is no standard way of writing a requirements specifications document, but here are a few **guidelines**:

SRS outline:

1. Purpose
2. Scope
3. System Overview
4. References
5. Definitions
6. Use Cases
7. Functional requirements
8. Non-functional requirements

Make things visual, a picture can save 1000 words. Graphics such as tables and charts should be included to communicate your ideas better.

Don't over-document:

SRS documents may get a bit long, avoid including things that may not need to be documented.

Keep an online version of the SRS and keep updating
as the tasks progress and if the staff and process changes, the SRS will be updated. For this reason, keeping a virtual version will help keep the whole team updated every time a change is made.

TESTING

Software development isn't an explicit science, and humans being as error prone as they're, code development should be amid quality assurance activities. It’s typical for developers to pay around four-hundredth of the entire project time on testing. For lifecritical code (e.g. control, reactor monitoring), testing will price three to five times the maximum amount as all alternative activities combined. The harmful nature of testing needs that the developer discard create mentally notions of the correctness of his/her developed code. this implies that testing should be done from a wholly totally different perspective from that of a developer.

In code development project, errors are available in at any stage throughout development. The most causes of errors are:

1. Not getting the proper needs,

2. Not obtaining the requirements right, and

3. Not translating the requirements in a very clear and understandable manner so programmers implement them properly.

Testing is a complete method. For testing we require 2 kinds of inputs. 1st is software configuration. It includes software requirement specification, design specifications and source code of program. Second is test configuration. It's essentially test plan and procedure.

Software configuration is needed in order that the testers know what's to be expected and tested whereas test configuration is testing plan that's, the method how the testing is going to be conducted on the system. It specifies the test cases and their expected value. It additionally specifies if any tools for testing are to be used. Test cases are needed to understand what specific things ought to be tested. Once tests are evaluated, test results are compared with actual results and if there's some error, then debugging is done to correct the error. Testing is a way to realize quality and reliability. Error rate that's the occurrence of errors is evaluated. This data can be used to predict the occurrence of errors in future.

BLACK BOX TESTING

Black box testing tests the overall practical requirements of product. Inputs are provided to product and outputs are verified. If the outputs obtained are same as the expected ones then the product meets the practical needs. during this approach internal procedures aren't considered. it's conducted at later stages of testing. now we willlook at black box testing technique.

Black box testing uncovers following kinds of errors.

1. Incorrect or missing functions

2. Interface errors

3. External database access

4. Performance errors

5. initialisation and termination errors.

WHITE BOX TESTING

In white box testing the internal functioning of the product is tested. every procedure is tested for its accuracy. it'smore intensive than black box testing. but for the overall product each of these techniques are crucial.

White box testing focuses on the internal functioning of the product. For this different procedures are tested. White box testing tests the subsequent

• Loops of the procedure

• Decision points

• Execution paths

Software Testing Levels

Unit Testing

A level of the software testing method where individual units/components of a software/system are tested. the aim is to validate that every unit of the software performs as designed.

Integration Testing

A level of the software testing method where individual units are combined and tested as a group. the aim of this level of testing is to reveal faults within the interaction between integrated units.

System Testing

A level of the software testing method where an entire, integrated system/software is tested. the aim of this test is to evaluate the system's compliance with the desired requirements.

Acceptance Testing

A level of the software testing method where a system is tested for acceptability. the aim of this test is to evaluatethe system's compliance with the business needs and assess whether or not it's acceptable for delivery.

SOFTWARE PROJECT MANAGEMENT

Software project management includes of:

– Project planning;

– Project scheduling;

– Risk management;

– Managing individuals.

Project Planning

•The biggest single downside that afflicts software developing is that of underestimating resources needed for a project.

•Developing a practical project plan is crucial to realize an understanding of the resources needed, and the waythese ought to be applied.

Types of plan:

– **Software development plan**. The central plan, that describes how the system is going to be developed.

– **Quality assurance plan**. Specifies the quality procedures & standards to be used.

– **Validation plan**. Defines how a consumer will validate the system that has been developed.

– **Configuration management plan**. Defines how the system is going to be configured and installed.

– **Maintenance plan**. Defines how the system is going to be maintained.

– **Staff development plan**. Describes how the abilities of the participants are going to be developed.

Work Breakdown

There are many ways of breaking down the activities in a project, however the most usual is into:

– Work packages;

– Tasks;

– Deliverables;

– Milestones.

•A **workpackage** is a large, logically distinct section of work:

– usually a minimum of twelve months duration;

– may include multiple simultaneous activities;

– independent of other activities;

– but may rely on, or feed into different activities;

– generally allocated to one team.

•A **task** is usually a much smaller piece of work: a section of a workpackage.

– usually 3–6 people’s month effort;

– may be dependent on other simultaneous activities;

– generally allocated to one person.

•A **deliverable** is an output of the project that can meaningfully be assessed.

Examples:

– A report (e.g., requirements spec);

– Code (e.g., alpha tested product). Deliverables are indicators (but only indicators) of progress.

•A **milestone** is a point at which progress on the project is assessed. generally a significant turning point within the project.

EXAMPLES:

– Delivery of requirements spec;

– Delivery of alpha tested code

Risks

When planning a project, it's critically vital to know what the key risks are, and is feasible plan for them:

•staff turnover;

•management change;

•hardware unavailability;

•requirements change;

•specification delays;

•size underestimate;

•technology change;

•Product competition.