International software failures

Data Loss at Gitlab

Two years ago a well-known code collaboration platform GitLab experienced a severe data loss which appeared to be one of the major outages in the IT world. GitLab originally used only one database server but decided to test a solution using two servers. They planned to copy the data from the production environment to the test environment.

In the process, the automatic mechanisms began to remove accounts from the database which were identified as dangerous. As a result of increased traffic, the data copying process began to slow down and then stopped completely due to data discrepancies. To add insult to injury, information from the production database was removed during the copying process.

After several attempts to resume the process, one of the employees decided to delete the test base and start the process again but accidentally deleted the production base. What made things even worse is that the directory holding the copies was empty too — the backups had not been made for a long time due to a configuration error.

What meant to be a standard procedure resulted in an 18-hour outage while the 300 GB of customer data was lost. According to the GitLab's estimates, the company has lost data on at least 5,000 new projects, 5,000 comments, and 700 users. The company approach to this failure deserves respect.

Gitlab explained in detail what happened, broadcasted the restoration procedure on YouTube and published a list of improvements to ensure that this trouble would never happen again. But as they say — the damage is done.

Google Plus Security Glitch

A vulnerability in Google+ exposed the private information of nearly 500 000 people using the social network between 2015 and March 2018. According to a report by the Wall Street Journal, the major part of the problem was a specific API that might be used to get access to non-public information.

The software glitch allowed outside developers to see the name, email address, employment status, gender, and age of the network's users. The error had been discovered in March 2018 and rectified immediately.

The interesting part is — Google did not share the information about the bug in Google+ at once trying not to get into the limelight of the Cambridge Analytica scandal and become noticed by the regulators.

At the same time, the WSJ report states, although Google has no evidence of data misuse it also can't say there was none. In any case, the tech backlash ended sadly for Google+ – the consumer version of the network was shut down shortly afterward.

British Airways "Technical Issue"

This summer the flag carrier airline of the UK — British Airways — reported an IT system issue that resulted in the delay of hundreds of flights in the UK, while dozens of flights were canceled completely. This failure affected three British airports and thousands of passengers who had to rebook their flights or check-in by using manual systems. Despite the problem being solved, the airports still felt the effect of this failure for a long while before normal service was resumed.

This computer problem at British Airways is just the latest in a series of IT concerns of the airline. Last year British Airways was sentenced to a record fine of 200 million euros for a data breach. **This happened because of the cyber-hack which resulted in a website failure compromising the data of 500 thousand customers. British Airways also experienced a massive system failure in 2017, which affected 75,000 passengers and cost the company nearly 80 million pounds**.

British Airways is not the only airline that is struggling with programming issues. In 2013 American Airlines had to ground off all its flights because of the computer glitch. And in 2017 the company had over 1,000 flights at risk of cancellation. The plans of many travelers during the holiday season could be ruined because of a single error in the company's internal scheduling system which gave too many pilots a day off.

Software Failures In Ghana

OLX

A classified forum which saw a decline of interest by people with the uprise of social media advertisement. People preferred to advertise on social media as it was less expensive and they have control over it. It was eventually bought by JiJi. Over the years, **OLX became a common place for fraud, especially sellers who post fake advertisements to dupe buyers upon receiving advance payment, and fraudulent buyers who engage in UPI scam, phishing and sending fake SMSes and emails of bank transfer confirmation**.

Easy Taxi

Easy Taxi was one of the first ride hailing apps that folded up months after entering the market, **It was new to the Ghanaian Market as most Ghanaians weren't accustomed to this service and to top it up, it was very expensive compared to the** traditional taxis that were already known. They also collapsed in Nigeria around the same period of their fold up in Ghana.

OMGvoice

OMGvoice became rather popular in their early days with creative content and young staff but folded up due to unrealistic targets. Techmoran.com reported how the "buzzfeed of Africa" had one of its founders resigning with lots of unpaid salaries due to the struggling nature of the company eventually leading to its collapse.

Most of the brands cited above did not only operate and fail in Ghana but in other African Countries like Nigeria and Kenya.
Online payment companies have not been left out either. Some of these companies that used to be very vibrant during their launch stage have been very quiet over the years. Examples of such are Slydepay and Zeepay who have been silent for some time now. With that said, it cannot be ignored that some of these online platforms have remained and continued to flourish, an example is ExpressPay.

Five Ws and one H

Application of the first W – ''What"

What do the users do? What are the objectives they have to achieve? You have to understand their tasks all in all and also the assignments that relate to the software testing or framework that you're planning.

Application of the second W 'Why'

The task team may ask 'Why' questions to get a more granular comprehension of the problem and look to clarify triggers or drivers that may have added to the problem. A portion of the questions for software testing that can be asked are:

Application of the third W "When"

By utilizing 'When' questions, project teams can time stamp the events and comprehend the connections among different events that may have affected the rise of the incompatibility problem

Application of the fourth W "Who"

Who are the users? What are their characteristics? What learning and experience do they convey to their tasks? Are there any other groups of users? Assuming this is the case, what separates them from one another, and which client bunches are generally vital? The subject of whose desires to meet dependably emerge. Would it be those of the Developers or of the Users? Users and engineers have their very own assumptions regarding the application and the codes. The desires from the two sides during software testing ought to be weighted legitimately. The project team can make questions to distinguish the people involved in contributing to the particular problem. A few questions that might be inquired:

Application of the fifth W "Where"

By asking 'Where' questions, the project team can improve the handle of the source(s) of the problem. A few questions that can be inquired during this stage of software testing are:

Now Application of the H "How"

In the metric reporting of 5W and 1H, anything that disappoints the client is a defect, thus understanding the client and client prerequisites is the most critical problem in building up a metric reporting software testing culture. This is a critical thinking administration philosophy that can be connected to a business procedure to recognize

and take out the main drivers of defects within software development, eventually enhancing the key software features and sparing expense for the association. In such a manner, the fundamental objective of metrics reporting is that any software development in an association should be monetarily suitable

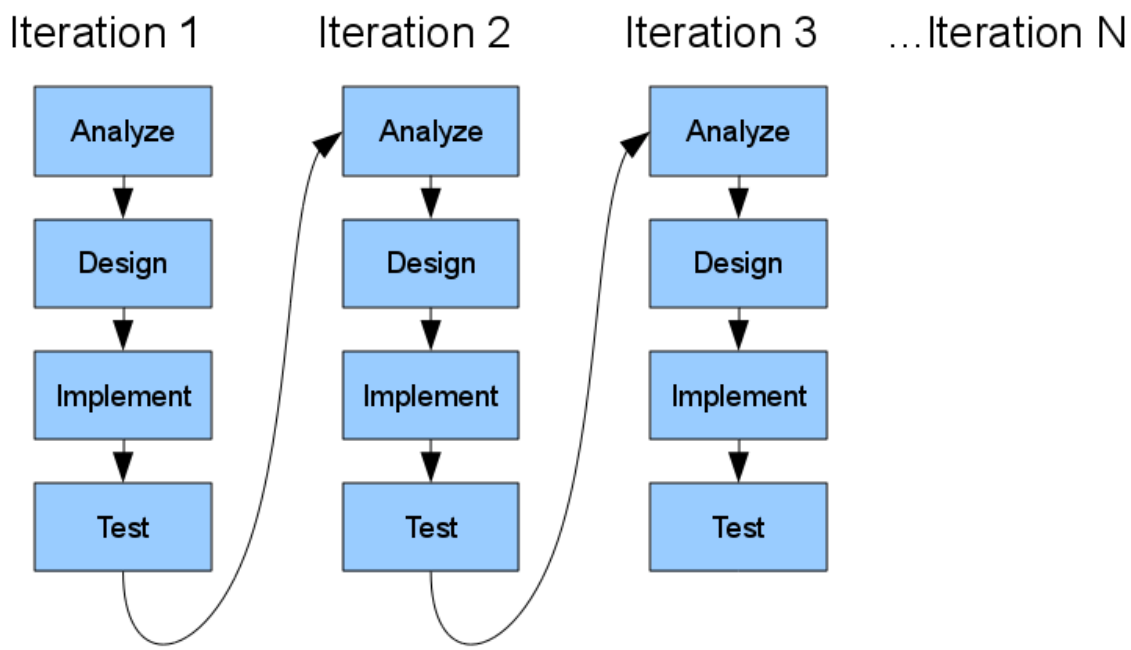
Two Myths of Software Engineering

Product must be perfect at the first attempt – it is an inherently a false believe. Nobody knows what 'perfect' looks like till people start using the product. Its better to build a Minimum Variable Product MVP with the essential functionality rather than require (and wait for) the nine yards from the developers. Put the MVP on the market, start earning money, take feedback and improve the product.

Remote Developers are worse than in-house developers – some clients believe than if developers are out of sight, they're unmotivated, uncontrollable, and unaccountable. This is hardly true, having modern project management practices, excellent communication opportunities and project management systems.

Two software engineering process models

Agile Model



An iterative approach. During each iteration a single feature or small set of features are chosen and implemented completely.

Advantages:

Can adapt to changing requirements because you haven't committed to big design that encompasses everything.

Easy to change direction to adapt to dynamic market conditions.

Disadvantages:

Used as an excuse for hacking - proceeding without a plan.

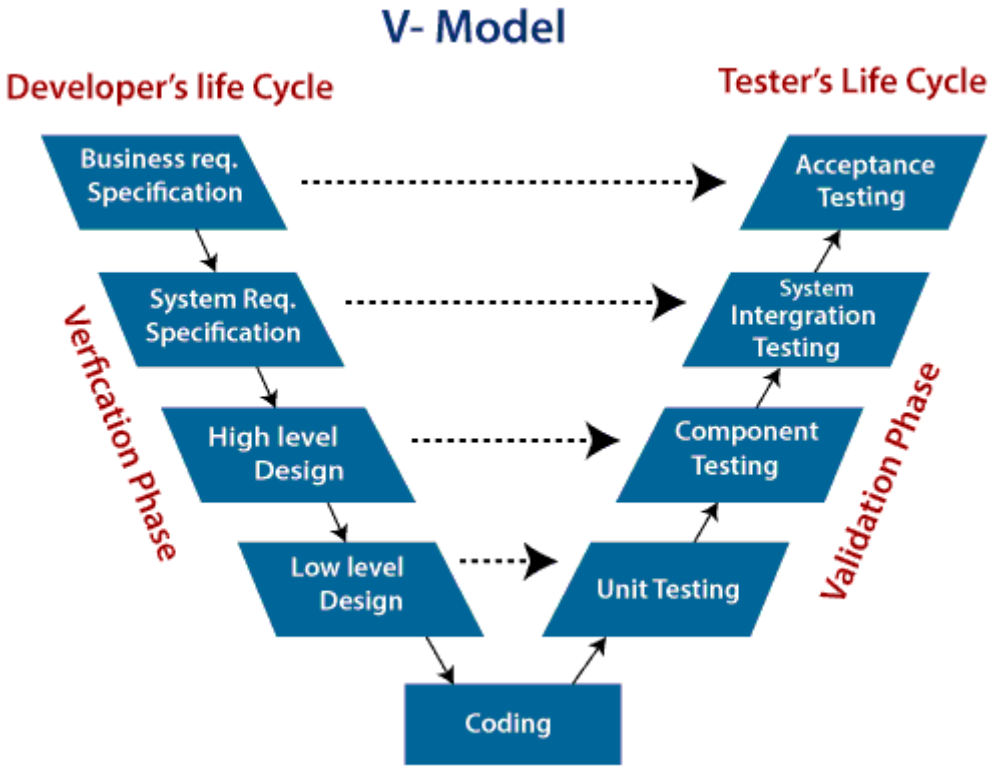
Substantial refactoring or redesign may be needed between iterations.

Not suitable for large projects or large teams.

Requires huge customer involvement, which is unusual to find.

V-Model

V-Model also referred to as the Verification and Validation Model. In this, each phase of SDLC must complete before the next phase starts. It follows a sequential design process same as the waterfall model. Testing of the device is planned in parallel with a corresponding stage of development.



Verification: It involves a static analysis method (review) done without executing code. It is the process of evaluation of the product development process to find whether specified requirements meet.

Validation: It involves dynamic analysis method (functional, non-functional), testing is done by executing code. Validation is the process to classify the software after the completion of the development process to determine whether the software meets the customer expectations and requirementsSo V-Model contains Verification phases on one side of the Validation phases on the other side. Verification and Validation process is joined by coding phase in V-shape. Thus it is known as V-Model.

There are the various phases of Verification Phase of V-model:

1. **Business requirement analysis:** This is the first step where product requirements understood from the customer's side. This phase contains detailed communication to understand customer's expectations and exact requirements.
2. **System Design:** In this stage system engineers analyze and interpret the business of the proposed system by studying the user requirements document.
3. **Architecture Design:** The baseline in selecting the architecture is that it should understand all which typically consists of the list of modules, brief functionality of each module, their interface relationships, dependencies, database tables,

architecture diagrams, technology detail, etc. The integration testing model is carried out in a particular phase.

4. **Module Design:** In the module design phase, the system breaks down into small modules. The detailed design of the modules is specified, which is known as Low-Level Design
5. **Coding Phase:** After designing, the coding phase is started. Based on the requirements, a suitable programming language is decided. There are some guidelines and standards for coding. Before checking in the repository, the final build is optimized for better performance, and the code goes through many code reviews to check the performance.

There are the various phases of Validation Phase of V-model:

1. **Unit Testing:** In the V-Model, Unit Test Plans (UTPs) are developed during the module design phase. These UTPs are executed to eliminate errors at code level or unit level. A unit is the smallest entity which can independently exist, e.g., a program module. Unit testing verifies that the smallest entity can function correctly when isolated from the rest of the codes/ units.
2. **Integration Testing:** Integration Test Plans are developed during the Architectural Design Phase. These tests verify that groups created and tested independently can coexist and communicate among themselves.
3. **System Testing:** System Tests Plans are developed during System Design Phase. Unlike Unit and Integration Test Plans, System Tests Plans are composed by the client?s business team. System Test ensures that expectations from an application developer are met.
4. **Acceptance Testing:** Acceptance testing is related to the business requirement analysis part. It includes testing the software product in user atmosphere. Acceptance tests reveal the compatibility problems with the different systems, which is available within the user atmosphere. It conjointly discovers the non-functional problems like load and performance defects within the real user atmosphere.

When to use V-Model?

- When the requirement is well defined and not ambiguous.
- The V-shaped model should be used for small to medium-sized projects where requirements are clearly defined and fixed.
- The V-shaped model should be chosen when sample technical resources are available with essential technical expertise.

Advantage (Pros) of V-Model:

1. Easy to Understand.
2. Testing Methods like planning, test designing happens well before coding.
3. This saves a lot of time. Hence a higher chance of success over the waterfall model.
4. Avoids the downward flow of the defects.
5. Works well for small plans where requirements are easily understood.

Disadvantage (Cons) of V-Model:

1. Very rigid and least flexible.
2. Not a good for a complex project.
3. Software is developed during the implementation stage, so no early prototypes of the software are produced.
4. If any changes happen in the midway, then the test documents along with the required documents, has to be updated.

Deliverables of Software Development Life Cycle

1. The first step in the SDLC

The first phase of SDLC is requirement analysis. The first phase includes collection of all the data from the customer. This includes the expectations of the customer. An understanding of what the product is, who the target audience are, why the product is being built is considered. Once the requirements are gathered, they are analysed. An analysis of how feasible the creation will be is made. Any further ambiguity is discussed. Once the requirement is understood clearly and the analysis made, the SRS (Software Requirement Specification), is created. This document is for the benefit of both the software developers and the customer. It can be referred to by both parties for convenience.

1. INTRODUCTION

1.1 PURPOSE

1.2 DOCUMENT CONVENTIONS

1.3 INTENDED AUDIENCE AND READING SUGGESTIONS

1.4 PROJECT SCOPE

1.5 REFERENCES

2. OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

2.2 PRODUCT FEATURES

2.3 USER CLASS and CHARACTERISTICS

2.4 OPERATING ENVIRONMENT

2.5 DESIGN and IMPLEMENTATION CONSTRAINTS

2.6 ASSUMPTION DEPENDENCIES

3. SYSTEM FEATURES

4. EXTERNAL INTERFACE REQUIREMENTS

4.1 USER INTERFACES

4.2 HARDWARE INTERFACES

4.3 SOFTWARE INTERFACES

4.4 COMMUNICATION INTERFACES

5. NONFUNCTIONAL REQUIREMENTS

5.1 PERFORMANCE REQUIREMENTS

5.2 SAFETY REQUIREMENTS

5.3 SECURITY REQUIREMENTS

5.4 SOFTWARE QUALITY ATTRIBUTES

2. The second step in the SDLC process

   The second step is designing. Once the requirements are understood, the software developers can move onto designing the software. The SRS document is kept as reference material while designing the software. All the agreements made in the SRS document are turned into a plan called Design specification. The team of software developers as well as the team of the customer reviews this. Any feedback or suggestion from either party is taken into consideration at this point. It is of utmost importance to have inputs from the customers at this stage. Failing at this stage will ensure either over expenditure or overall failure of the software

   - Introduction
   - Document Outline
   - Document Description
     1. Introduction
     2. System Overview
     3. Design Considerations
        - Assumptions and Dependencies
        - General Constraints
        - Goals and Guidelines
        - Development Methods
     4. Architectural Strategies
     5. System Architecture
        - Subsystem Architecture

6. Policies and Tactics
7. Detailed System Design
    - Detailed Subsystem Design
8. Glossary
9. Bibliography

3. The Third Phase

The Design phase is followed by the Implementation and Coding Phase. Here, the different methodologies come into play. A choice can be made between the agile, waterfall or any other more suitable method. All the planning and decisions have been finalised at this point and a blueprint is drawn up. The actual implementation of the ideas in the blueprint is done in this stage of SDLC. Tasks are divided into modules to be distributed among the developers according to their strengths. This is the longest phase of the SDLC life cycle. It is very important to keep the customers involved in every step of the way to avoid any kind of mis-communication. The document produced at this stage is the code review document.

The Fourth Phase

TABLE OF CONTENTS

By: **Philip Nyarko**

(Phil)