Assignment

Day 1

Sunday, 3 September 2023

By: Ibrahim Tarek

SV G2 / Intake #3

Table of Contents

1. Executive Summary

This report discusses each of the following topics:

1- What is SDLC (Software Development Life Cycle).
2- What is the difference between wild pointer and dangling pointer.

2. SDLC

The Software Development Life Cycle (SDLC) refers to a methodology with clearly defined processes for creating high-quality software. in detail, the SDLC methodology focuses on the following phases of software development:

- Requirement analysis
- Planning
- Software design such as architectural design
- Software development
- Testing
- Deployment
- Maintenance

This report will explain how SDLC works, dive deeper in each of the phases, and provide you with examples to get a better understanding of each phase.

2.1. Definition of SDLC

SDLC or the Software Development Life Cycle is a process that produces software with the highest quality and lowest cost in the shortest time possible. SDLC provides a well-structured flow of phases that help an organization to quickly produce high-quality software which is well-tested and ready for production use.

The SDLC involves seven phases as explained in the introduction. Common SDLC models include:

- Waterfall Model.
- V-Shaped Model.
- Iterative Model.
- Spiral Model.
- Big Bang Model.
- Agile Model.

2.2. Importance of SDLC

Here are a few benefits the SDLC can bring:

- Better visibility of a project plan.
- Improved control and monitoring.
- Project risk mitigation and error reduction.
- A method of project progress tracking.
- Timely delivery of high-quality software to client.

2.3. SDLC Stages

2.3.1. Requirement Analysis

This is the first and fundamental stage of SDLC. Business analysts gather requirements from their customers, target market, and industry experts to create a Business Specification (BS) document. Other organizations and teams may refer to this document as Customer Requirement Specification (CRS).

The intent of this document is to outline current pain points that developers should strive to eliminate through the power of software. It can be a useful resource to help the team discover innovative methods to change and improve their products.

The document is shared with the development team, which then uses it in the next stage.

2.3.2. Planning

“What do we want?”

In this stage of the SDLC, the team determines the cost and resources required for implementing the analyzed requirements. It also details the risks involved and provides sub-plans for softening those risks.

In other words, the team should determine the feasibility of the project and how they can implement the project successfully with the lowest risk in mind.

2.3.3. Design

“How will we get what we want?”.

This stage focuses on designing the product. It involves product architects and developers who will ideate and present a design of the product. They may present more than one design approach, and these ideas are documented in a Design Document Specification (DDS).

The design phase is where you put pen to paper—so to speak. The original plan and vision are elaborated into a software design document (SDD) that includes the system design, programming language, templates, platform to use, and application security measures. This is also where you can flowchart how the software responds to user actions.

In most cases, the design phase will include the development of a prototype model. Creating a pre-production version of the product can give the team the opportunity to visualize what the product will look like and make changes without having to go through the hassle of rewriting code.

2.3.4. Software Development

“Let’s create what we want.”

The actual development phase is where the development team members divide the project into software modules and turn the software requirement into code that makes the product.

This SDLC phase can take quite a lot of time and specialized development tools. It’s important to have a set timeline and milestones so the software developers understand the expectations and you can keep track of the progress in this stage.

In some cases, the development stage can also merge with the testing stage where certain tests are run to ensure there are no critical bugs.

2.3.5. Testing

“Did we get what we want?”

Before getting the software product out the door to the production environment, it’s important to have your quality assurance team perform validation testing to make sure it is functioning properly and does what it’s meant to do. The testing process can also help hash out any major user experience issues and security issues.

In some cases, software testing can be done in a simulated environment. Other simpler tests can also be automated.

The types of testing to do in this phase:

- **Performance testing:** Assesses the software's speed and scalability under different conditions
- **Functional testing:** Verifies that the software meets the requirements
- **Security testing:** Identifies potential vulnerabilities and weaknesses
- **Unit-testing:** Tests individual units or components of the software
- **Usability testing:** Evaluates the software's user interface and overall user experience
- **Acceptance testing:** Also termed end-user testing, beta testing, application testing, or field testing, this is the final testing stage to test if the software product delivers on what it promises.

2.3.6. Deployment

“Let’s start using what we got.”

At this stage, the goal is to deploy the software to the production environment so users can start using the product. However, many organizations choose to move the product through different deployment environments such as a testing or staging environment.

This allows any stakeholders to safely play with the product before releasing it to the market. Besides, this allows any final mistakes to be caught before releasing the product.

2.3.7. Maintenance

"Let's get this closer to what we want."

In the maintenance stage, users may find bugs and errors that were missed in the earlier testing phase. These bugs need to be fixed for better user experience and retention. In some cases, these can lead to going back to the first step of the software development life cycle.

The SDLC phases can also restart for any new features you may want to add in your next release/update.

2.4. SDLC Common Models

2.4.1. Waterfall Model

Waterfall model is the very first model that is used in SDLC. It is also known as the linear sequential model.

In this model, the outcome of one phase is the input for the next phase. Development of the next phase starts only when the previous phase is complete.

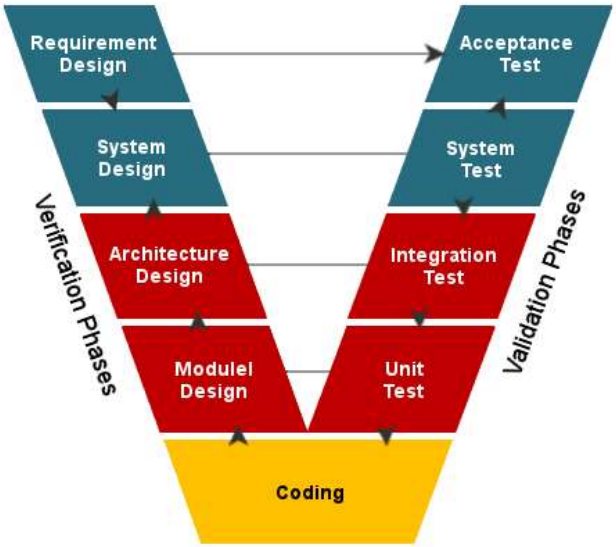
Advantages of the Waterfall Model:

- Waterfall model is the simple model which can be easily understood and is the one in which all the phases are done step by step.
- Deliverables of each phase are well defined, and this leads to no complexity and makes the project easily manageable.

Disadvantages of Waterfall model:

- Waterfall model is time-consuming & cannot be used in the short duration projects as in this model a new phase cannot be started until the ongoing phase is completed.
- Waterfall model cannot be used for the projects which have uncertain requirement or wherein the requirement keeps on changing as this model expects the requirement to be clear in the requirement gathering and analysis phase itself and any change in the later stages would lead to cost higher as the changes would be required in all the phases.

2.4.2. V-Shaped Model



V-Model is also known as Verification and Validation Model. In this model Verification & Validation goes hand in hand i.e. development and testing goes parallel. V model and waterfall model are the same except that the test planning and testing start at an early stage in V-Model.

Advantages of V – Model:

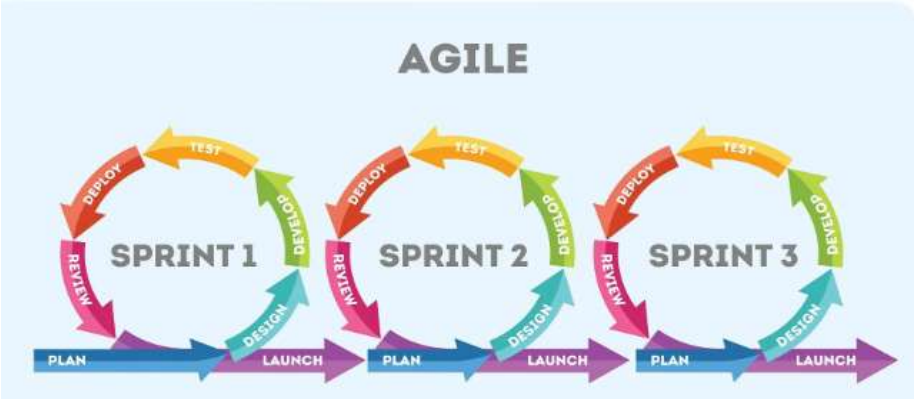
- Proactive defect tracking – that is defects are found at early stage.
- Avoids the downward flow of the defects.
- V –model approach is good for smaller projects wherein the requirement is defined and it freezes in the early stage.
- It is a systematic and disciplined model which results in a high-quality product.

Disadvantages of V-model:

- Very rigid and least flexible.
- Software is developed during the implementation phase, so no early prototypes of the software are produced.
- V-shaped model is not good for ongoing projects.
- Requirement change at the later stage would cost too high.

2.4.3. Agile Model

This model focuses more on flexibility while developing a product rather than on the requirement.



In Agile, a product is broken into small incremental builds. It is not developed as a complete product in one go.

Each build increments in terms of features. The next build is built on previous functionality.

In agile iterations are termed as sprints. Each sprint lasts for2-4 weeks. At the end of each sprint, the product owner verifies the product and after his approval, it is delivered to the customer. Customer feedback is taken for improvement and his suggestions and enhancement are worked on in the next sprint. Testing is done in each sprint to minimize the risk of any failures.

Advantages of Agile Model:

- It allows more flexibility to adapt to the changes.
- The new feature can be added easily.
- Customer satisfaction as the feedback and suggestions are taken at every stage.

Disadvantages:

- Lack of documentation.
- Agile needs experienced and highly skilled resources.
- If a customer is not clear about how exactly they want the product to be, then the project would fail.

2.5. Which SDLC model is right for you?

Each of these SDLC methodologies offers unique process for the variety of project challenges you will encounter in your career.

Finding the right one depends heavily on not just the expected outcome, but the parameters by which the project is executed.

3. Dangling Pointer vs Wild Pointer

3.1. Dangling Pointer

A pointer pointing to a memory location that has been deleted (or freed) is called a dangling pointer. There are three different ways where Pointer acts as a dangling pointer (explanation code is in C):

3.1.1. By deallocating memory

```
void main(void)
{
   int *ptr = (int *) malloc(sizeof(int));

   // After below free call, ptr becomes a dangling pointer
   free(ptr);
}
```

3.1.2. Function Call

```
int *fun()
{
   // x is local variable and goes out of scope after an execution of fun() is over.
   int x = 5;

   return &x;
}

Void main(void)
{
   // p points to something which is not valid anymore
   int *p = fun();
}
```

3.1.3. When the variable goes out of scope

```
void main()
{
  int *ptr;

  {
     int x;
     ptr = &x;
  }

  // Here ptr is dangling pointer
}
```

3.2. Wild Pointer

Wild Pointers are the Uninitialized pointers because they point to some arbitrary memory location and may cause a program to crash or behave unexpectedly.

3.2.1. Explanation Code in C

```
int main()
{
   // here p is a wild pointer
   int* p;
   int a = 10;
   // now p is not a wild pointer
   p = &a;
}
```

4. References

4.1. https://stackify.com/what-is-sdlc/

4.2. https://www.roberthalf.com.au/blog/employers/6-basic-sdlc-methodologies-which-one-best

4.3. https://theproductmanager.com/topics/software-development-life-cycle/

4.4. https://www.softwaretestinghelp.com/software-development-life-cycle-sdlc/

4.5. https://www.geeksforgeeks.org/dangling-void-null-wild-pointers/