Capstone Design in CSE
컴퓨터공학 종합설계

2. SW Development Life Cycle

Quick Review of Software Development Process

- Source
  - http://www.tutorialspoint.com/software_engineering/
  - Goodrich, Tamassia, and Mount, Data structures and algorithms, 2nd edition
  - https://en.wikipedia.org/wiki/Agile_software_development

Definitions

- Software
  - collection of executable programming code, associated libraries and documentations.
  - Software, when made for a specific requirement is called software product.
- Engineering
  - developing products, using well-defined, scientific principles and methods.
- Software Engineering
  - an engineering branch associated with development of software product using well-defined scientific principles, methods and procedures.
  - The outcome of software engineering is an efficient and reliable software product.

Three categories of software

- **S-type (static-type)**
  - This is a software, which works strictly according to **defined specifications and solutions**. The solution and the method to achieve it, both are immediately understood before coding. The s-type software is least subjected to changes hence this is the simplest of all.
  - For example, **calculator program** for mathematical computation.
- **P-type (practical-type)**
  - This is a software with a collection of procedures. This is defined by exactly what procedures can do. In this software, the specifications can be described but the solution is not obvious instantly.
  - For example, **gaming software.**
- **E-type (embedded-type)**
  - This software works closely as the **requirement of real-world environment**. This software has a high degree of evolution as there are various changes in laws, taxes etc. in the real world situations.
  - For example, **Online trading software**.

Good Software

- Operational
  - This tells us how well software works in operations. It can be measured on:
    Budget / Usability / Efficiency / Correctness / Functionality / Dependability / Security / Safety
- Transitional
  - This aspect is important when the software is moved from one platform to another:
    Portability / Interoperability / Reusability / Adaptability
- Maintainable
  - This aspect briefs about how well a software has the capabilities to maintain itself in the ever-changing environment:
    Modularity / Flexibility / Scalability

Software Development Life Cycle (SDLC)

1. Communication
2. Requirement gathering
3. Feasibility study
4. System analysis
5. Software design
6. Coding
7. Testing
8. Integration
9. Implementation
10. Operation and Maintenance
11. Disposition

Software Development Life Cycle (SDLC)

1. Communication
2. Requirement gathering
3. Feasibility study
4. System analysis
5. Software design
6. Coding
7. Testing
8. Integration
9. Implementation
10. Operation and Maintenance
11. Disposition

This is the first step where the user initiates the request for a desired software product. He contacts the service provider and tries to negotiate the terms. He submits his request to the service providing organization **in writing.**

Software Development Life Cycle (SDLC)

1. Communication
2. Requirement gathering
3. Feasibility study
4. System analysis
5. Software design
6. Coding
7. Testing
8. Integration
9. Implementation
10. Operation and Maintenanc
11. Disposition

The software development team tries to bring out as much information as possible on the users' requirements.
The requirements are contemplated and segregated into
- user requirements,
- system requirements and
- functional requirements
by the following practices:
- studying the existing or obsolete system and software,
- conducting interviews of users and developers,
- referring to the database or
- collecting answers from the questionnaires

Software Development Life Cycle (SDLC)

1. Communication
2. Requirement gathering
3. Feasibility study
4. System analysis
5. Software design
6. Coding
7. Testing
8. Integration
9. Implementation
10. Operation and Maintenance
11. Disposition

Analyze if a software can be made to fulfill all requirements. See if the project is financially, practically and technologically feasible.

Software Development Life Cycle (SDLC)

1. Communication
2. Requirement gathering
3. Feasibility study
4. System analysis
5. Software design
6. Coding
7. Testing
8. Integration
9. Implementation
10. Operation and Maintenance
11. Disposition

The developers decide a **roadmap** of their plan and try to bring up the best software **model** suitable for the project. Analyzes the scope of the project and plans the **schedule and resources** accordingly.

Software Development Life Cycle (SDLC)

1. Communication
2. Requirement gathering
3. Feasibility study
4. System analysis
5. Software design
6. Coding
7. Testing
8. Integration
9. Implementation
10. Operation and Maintenance
11. Disposition

Bring down whole knowledge of requirements and analysis on the desk and design the software product.

Engineers produce

- meta-data and data dictionaries,
- logical diagrams,
- data-flow diagrams and
- in some cases pseudo codes.

Software Development Life Cycle (SDLC)

1. Communication
2. Requirement gathering
3. Feasibility study
4. System analysis
5. Software design
6. Coding
7. Testing
8. Integration
9. Implementation
10. Operation and Maintenance
11. Disposition

Also known as programming phase. The implementation of software design starts in terms of writing program code in the suitable programming language and developing **error-free** executable programs efficiently.

Software Development Life Cycle (SDLC)

1. Communication
2. Requirement gathering
3. Feasibility study
4. System analysis
5. Software design
6. Coding
7. Testing
8. Integration
9. Implementation
10. Operation and Maintena
11. Disposition

An estimate says that 50% of whole software development process should be tested. Software testing is done while coding by the developers and thorough testing is conducted by testing experts at various levels of code such as

- module testing,
- program testing,
- product testing,
- in-house testing and
- testing the product at user's end.

Early discovery of errors and their remedy is the key to reliable software.

Software Development Life Cycle (SDLC)

1. Communication
2. Requirement gathering
3. Feasibility study
4. System analysis
5. Software design
6. Coding
7. Testing
8. Integration
9. Implementation
10. Operation and Maintenance
11. Disposition

Software may need to be integrated with the libraries, databases and other program(s). This stage of SDLC is involved in the integration of software with outer world entities.

Software Development Life Cycle (SDLC)

1. Communication
2. Requirement gathering
3. Feasibility study
4. System analysis
5. Software design
6. Coding
7. Testing
8. Integration
9. Implementation (Installation)
10. Operation and Maintenance
11. Disposition

This means installing the software on user machines. At times, software needs post-installation configurations at user end. Software is tested for portability and adaptability and integration related issues are solved during implementation.

Software Development Life Cycle (SDLC)

1. Communication
2. Requirement gathering
3. Feasibility study
4. System analysis
5. Software design
6. Coding
7. Testing
8. Integration
9. Implementation
10. Operation and Maintenance
11. Disposition

This phase confirms the software operation in terms of more efficiency and less errors.
The software is maintained timely by updating the code according to the changes taking place in user end environment or technology.
This phase may face challenges from hidden bugs and real-world unidentified problems.

Software Development Life Cycle (SDLC)

1. Communication
2. Requirement gathering
3. Feasibility study
4. System analysis
5. Software design
6. Coding
7. Testing
8. Integration
9. Implementation
10. Operation and intenar
11. Disposition

As time elapses, the software may decline on the performance front. It may go completely obsolete or may need intense upgradation. Hence a pressing need to eliminate a major portion of the system arises.

This phase includes archiving data and required software components, closing down the system, planning disposition activity and terminating system at appropriate end-of-system time.

Software Development Life Cycle (SDLC)

1. Communication
2. Requirement gathering
3. Feasibility study
4. System analysis
5. Software design
6. Coding
7. Testing
8. Integration
9. Implementation
10. Operation and Maintenance
11. Disposition

In all stages, **documentation** is necessary.

Software Development Paradigm

- Strategy to develop the software
- That is, how to develop?
- A few well-known paradigms
  1. Waterfall model
  2. Iterative model
  3. Spiral model
  4. V-model
  5. Big-bang model

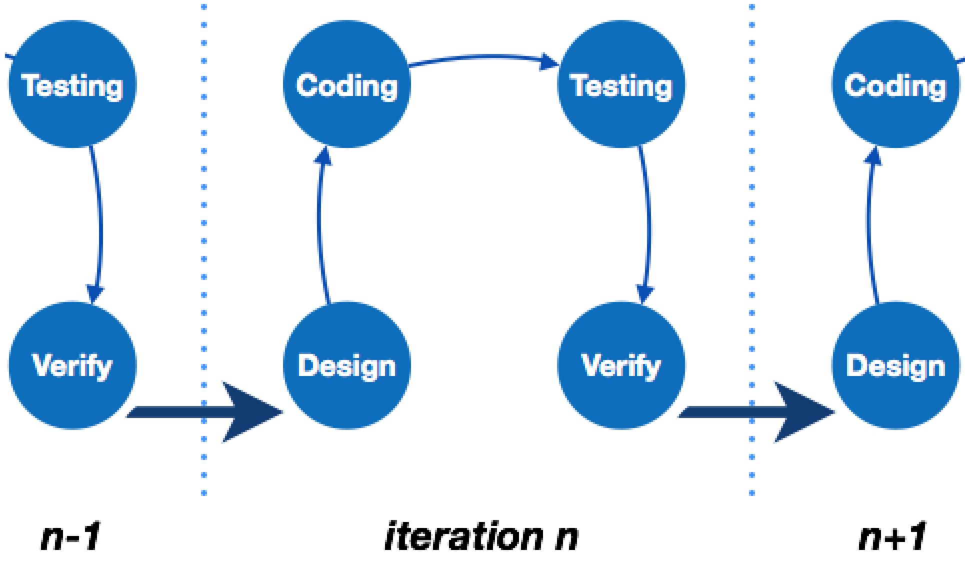
Software Development Paradigm
1. Waterfall model

- Simplest model
- All the phases of SDLC will function one after another in linear manner.
- Assumes that everything is carried out and taken place perfectly as planned in the previous stage.
- Best suited when developers already have designed and developed similar software in the past and are aware of all its domains.

1. Communication
2. Requirement gathering
3. Feasibility study
4. System analysis
5. Software design
6. Coding
7. Testing
8. Integration
9. Implementation
10. Operation and Maintenance
11. Disposition

Software Development Paradigm 2. Iterative model

- The software is first developed on very small scale and all the steps are followed which are taken into consideration.
- Then, on every next iteration, more features and modules are designed, coded, tested and added to the software.
- Every cycle produces SW, which is complete in itself and has more features than the previous one.
- It is easier to manage the development process but it consumes more resources.



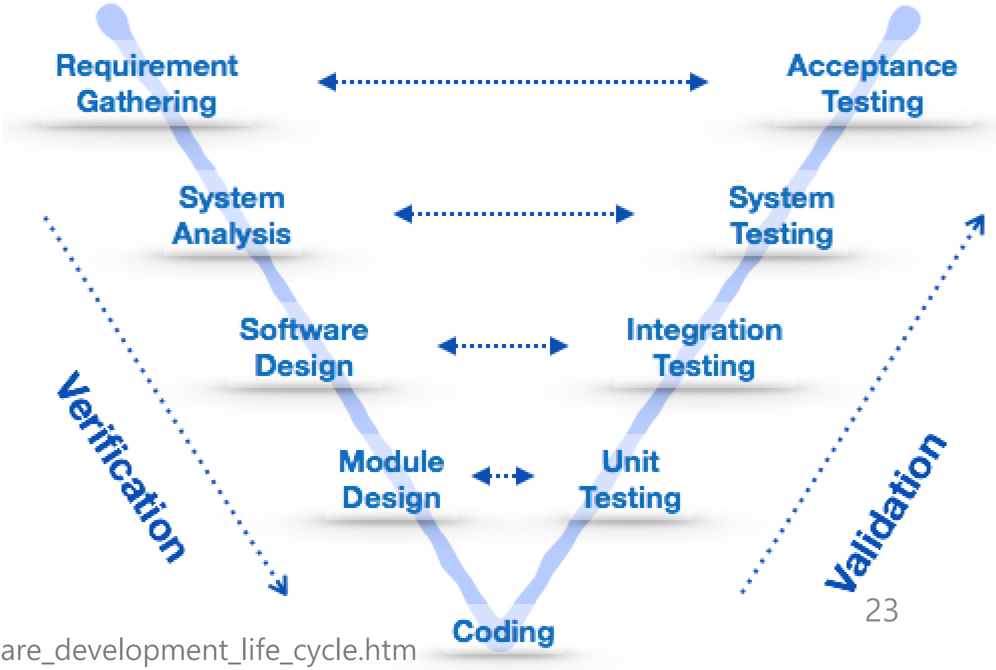
Source: http://www.tutorialspoint.com/software_engineering/software_development_life_cycle.htm

Software Development Paradigm
3. Spiral model

- Combination of the iterative model and another model
- You choose one SDLC model and combine it with cyclic process.

Software Development Paradigm 4. V model

- “V and V” (Verification and Validation) model
- The major drawback of waterfall model is we move to the next stage only when the previous one is finished and there was no chance to go back if something is found wrong in later stages.
- V-Model provides means of testing of software at each stage in reverse manner.



Source: http://www.tutorialspoint.com/software_engineering/software_development_life_cycle.htm

Software Development Paradigm 5. Big bang model

- conceptualized around the big bang of universe.
- requires little planning,
  lots of programming and lots of funds.
- It does not follow any process, or at times the customer is not sure about the requirements and future needs. So the input requirements are arbitrary.
- Not suitable for large software projects but good one for learning and experimenting.

New Approach

- Problem of previous (waterfall) model
  - It is not easy to separate each stage. It is not easy to start the next stage after the previous stage is fully complete.
  - No communication with users: No visual verification until (almost) last stage.
  - It consumes significant resources for documentation and management.
  - Not appropriate for SW development?

New Approach

- Agile software development
  - requirements and solutions **evolve** through the collaborative effort of self-organizing and cross-functional teams and their customer(s)/end user(s)
  - Iterative, incremental and evolutionary development
  - Efficient and face-to-face communication
    - early delivery
  - Very short feedback loop and adaptation cycle
    - continual improvement
    - rapid and flexible response to change
  - Less documentation and more coding
  - Many tools are available. (e.g., Jira)