Software Development

Dr. Hamada I. AbdulWakel[1]

[1]Computer Science Department

2023 - 2022

Ch2: Process Life Cycle Models

Software Development Life Cycle

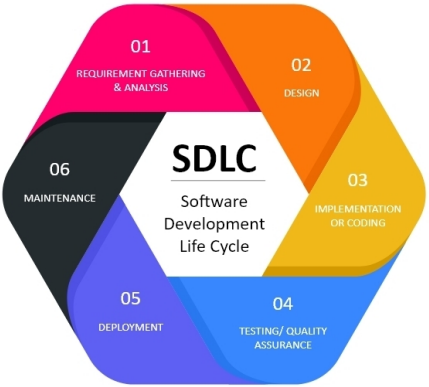
- Referred to as the SDLC.
- The general steps to build software.
- Each phase's team members have their responsibilities.
- The number of phases is varied depending on organization's needs.
- The names of the phases might not even be the same, but the ideas behind them are the same.

Cycle?!!!


The entire process will repeat for major releases
The process moves from step to step
There is a feedback loop to accommodate changes and or failures

Software Development Life Cycle

Methodology
A formalized approach that an organization goes through to analyse, design, implement, and maintain a system.



Phase 1: Requirement Gathering & Analysis

- Goal is to develop a clear understanding of **the new/needed** system's requirements.
- Business analysts & Project manager will build a complete requirements specification:
    1. Interviews.
    2. Study existing system and its problems.
    3. Identify improvements (i.e., Functional requirements and Non-Functional requirements).
    4. Sample screen designs & reports.
- These requirements are approved by the customer prior to design work.
- Go/No Go decision made by sponsor and steering committee.

Phase 2: Design

- All of the "logical" work is converted to the "physical".
- Software designers & Architects will map requirements to:
    1. Architecture design.
    2. High level design (HLD) such as modules and integration.
    3. Low level design (LLD) such as algorithms, CD, DFD, etc.
    4. Database design.
    5. API design.
    6. Infrastructure design.
- These designs will be reviewed by various departments for approval.
    1. Software development.
    2. Business & Customers.

Phase 3: Implementation

- Programmers begin working on writing code to meet business requirements.
- Programmers will look to reuse code from previous projects to save time.
- Unit testing should begin at this point.

Phase 4: Testing

- This is a critical step to ensure the system works properly.
- Testers should perform testing in a systematic way and document the results carefully.
- Systems may have multiple layers of testing:
    1. Unit testing.
    2. Integration testing.
    3. System testing.
    4. User acceptance test (UAT).

Phase 5: Deployment

- In this step software is release to the production environment for users to being operations.
- Project manager coordinates the deployment plan with all the stakeholders, and set the deployment date.
    1. Complex systems may be rolled out in phases.
    2. System outages may be required to perform software and hardware updates.

Phase 6: Maintenance

- Fix software bugs or issues that are identified during usage of the software application in the production environment.
- Support operations begin at this point to ensure smooth operation of the system:
  1. On-going system support to solve users problems and questions.
  2. Update and test the software programs and configurations to fix the identified issue/bug.

Question

Why there are Win 8, Win 8.1, and Win 10 and there is not Win 8.2 ?

Software Development Types

- **As-is-System:**

  In this system, the SDLC phases are applied to guide the project team to maintain or enhance the current system of the organization.

- **To-be-System:**

  In this system, the SDLC phased are applied to build a new system to the organization.

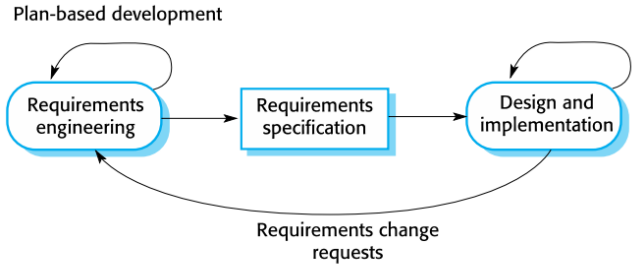
Software Development Models

- Code and fix.
- Plan-driven approaches.

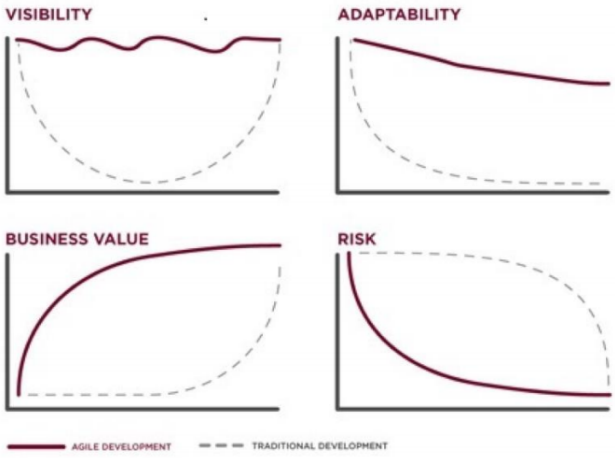
  Approaches where development processes are planned in advance and progress is measured against this plan.
- Agile approaches.

  Approaches where development processes planning is incremental and it is easier to change the process to reflect changing in customer requirements.

Software Development Models ...



Software Development Models ...

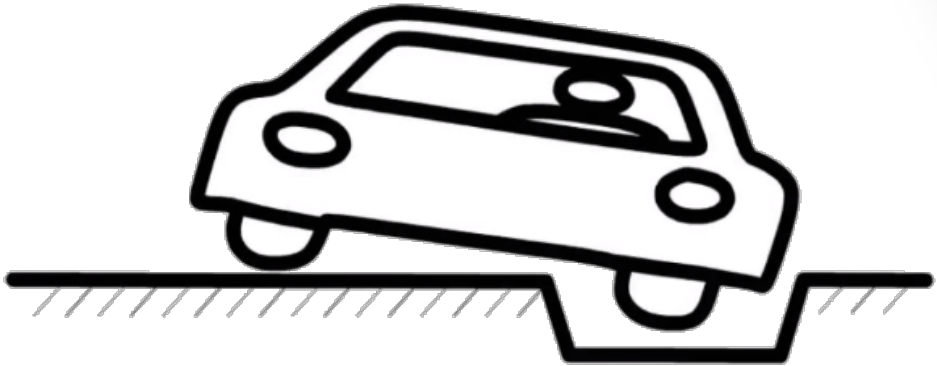


Code and Fix Model

Code and Fix Model

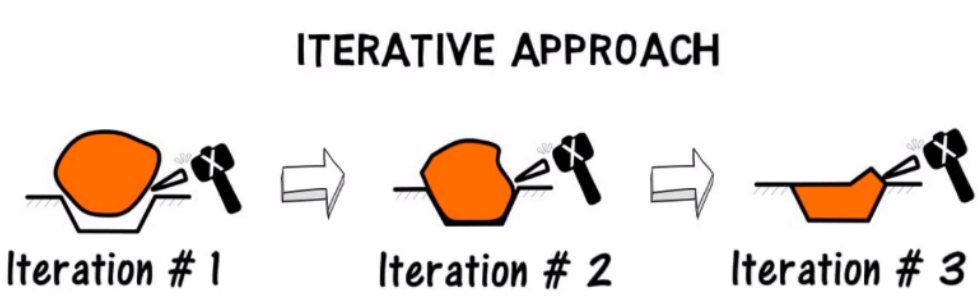
- Huge risk for client satisfaction and even failure as there is no concrete understanding of requirements.
- Time saver and is a convenient model for low-budget projects.
- Free of the formal instructions from the managerial stuff.
- No room for structured design.
- Small projects that won't require any further development.
- Suitable for small programming (100 or 200 lines).
- Not suitable for big engineering team.

Think and Solve

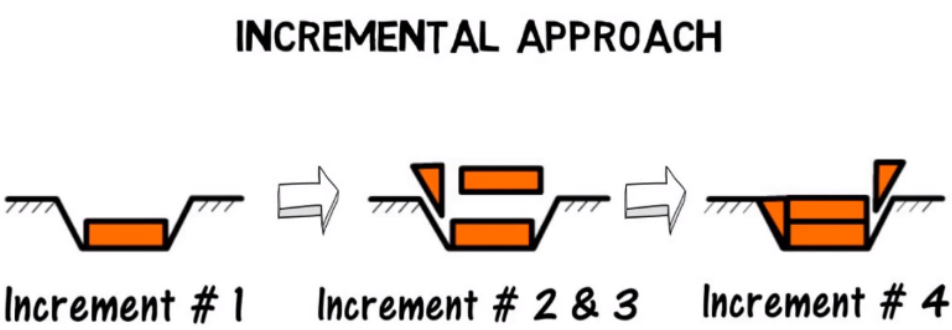


Solution Scenario 1

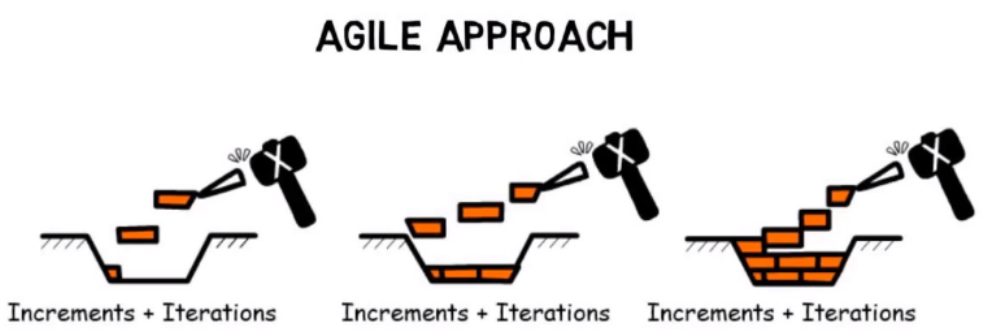
Solution Scenario 2



Solution Scenario 3



Solution Scenario 4



Plan-driven Models

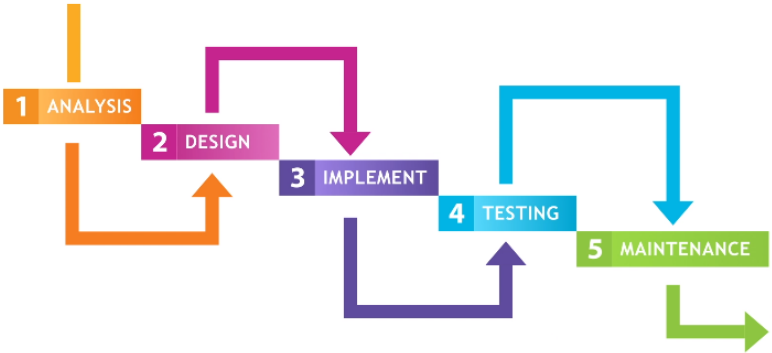
- All of the process activities are planned in advance and progress is measured against this plan.
- Plan-driven methods works best when developers can determine the requirement in advance, and when the requirements remain relatively stable, with change rates of 1%/month.
- Plan-driven often are better for newer/less-experienced programmers and large & distributed team members.
- Plan-driven is also good for projects with a lot of components that have to be integrated.

Plan Driven Characteristics

- Focus on repeatability and predictability.
- Defined, standardized, and incrementally improving processes.
- Detailed plans, workflow, roles, responsibilities, and work product descriptions (Documentation).
- A software system architecture defined up-front.
- On-going risk management.
- Focus on verification and validation. (Differences???)

The Waterfall Model

- A project management methodology based on a sequential (linear) design process.
- Getting the product right on the first try.
- Phases:



When to use Waterfall Model?

- Large-scope and long-term projects "Ex. Digital Egypt Platforms".
- Requirements are very well known and do not require change your mindset (i.e, small projects).
- Technology understood. Suitable when developers already have designed and developed similar software.
- New version of an existing product.
- Porting an existing product to a new platform.

The Waterfall Model Pros and Cons

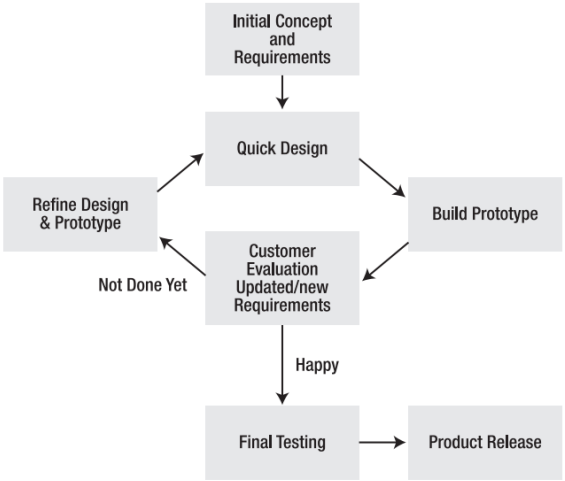
- Pros:
  1. Easy to understand and use (clarity on milestones).
  2. Provide structure to inexperienced staff.
  3. Works well when quality is more important than cost or schedule.
- Cons:
  1. The business requirements section is the most crucial aspect. All requirements must be known upfront.
  2. Not change friendly.
  3. Deliverables created for each phase are considered frozen/voluminous (i.e., inhibits flexibility).
  4. Little opportunity for customers to preview the system.
  5. It might be difficult to foresee all the risks which will come in later phases of development.
  6. Testing phase begin late and must to finish early. Quality only increased when all bugs are fixed.

The Waterfall Model at All

Evolutionary Prototyping Model

- While developing software, it is often needed to make modifications in earlier phase or task set.
- In such cases, iterative approach need to be adopted.
- Requirements are prioritized in a series of **feature-rich versions** of the software.
- Prototype??

Evolutionary Prototyping Model ...



Evolutionary Prototyping Model Pros and Cons

- Pros:
  1. Customer get early interaction with system.
  2. Missing functionality can be easily figured out.
  3. Customer and developer have a baseline to work together, therefore, the risk of implementing the wrong system is minimized.
  4. Suitable for dynamic and unclear requirements.
  5. Suitable for application domain that isn't well known.
- Cons:
  1. Quick prototypes with missing documentation lead to poor performance.
  2. Disrupts your schedule. Difficult to plan.
  3. Difficult to determine when you're completed because we don't know how many iterations will be needed.
  4. Some people say it similar to Fix and Code model, which lead to expensive and unplanned prototype iterations.