Systems development life cycle

Software Engineering I
Ben Mezger
https://seds.nl
10/04/2017

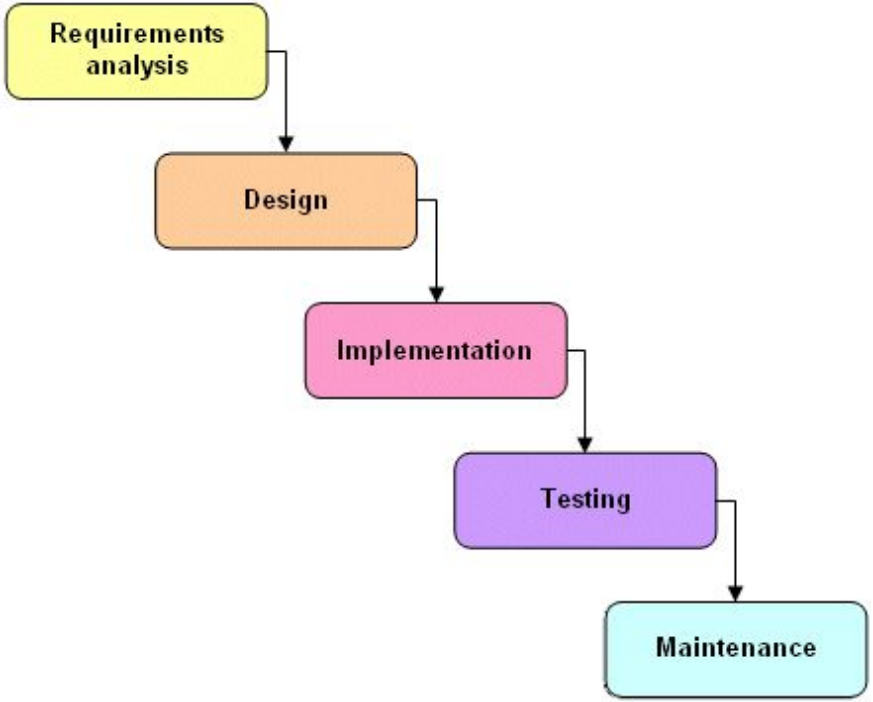
Introduction

- Systems development life cycle (SDLC) help making the decision and organization of a project;
- Helps organize and distribute activities;
- Some of the methodologies are:
  - Waterfall development;
  - Evolutionary prototyping;
  - Spiral development;
  - Iterative development;
  - Etc;

Waterfall

- Is a sequential development approach, based on phases and documentation;
- Each phase is structured in sub-activities that can be executed concurrently by different people;
  - Requirements -> Design -> Coding and module testing -> Integration and system testing -> delivery, deployment and maintenance;
- Different variations of the waterfall model exist;
  - Different variations can be based on criticality and complexity of the application;
- All different variation of the model insist that the *requirements* phase be completed before proceeding to the *design* phase;
  - The same follows for each phase;

Waterfall - Example



Waterfall - *DOD-STD-2167A*

- Department of defence standards;
  - Defence Systems Software development;
- Published on Feb. 29, 1988;
- “uniform requirements for the software development that are applicable throughout the system life cycle.”;
- Made of the applied on different use cases, with the exception of small applications that execute a fixed amount of functions that won’t change in the life-time of the system;

Critical evaluation of the Waterfall Model

- The waterfall model brought two fundamental contribution to Software Engineering;
  - Development process of Software required discipline, planning and management;
  - Implementation of the product must be postponed until all the objectives are completely understood;
- The waterfall model assumes that the development of Software follows linearly of the programming analysis - in practice, this does not happen;
  - *Beta* and *testing* versions provide a feedback about previous phases;

Waterfall - Advantages

- Allows departmentalization and control;
- Simple to use and understand;
- Easy to manage, since the model is very rigorous;
  - Each phase is very specific and easy to review in the process;
- Each phase is well defined;
- Easy to distribute tasks;
- Processes and results are well documented;
- Very clear milestones;

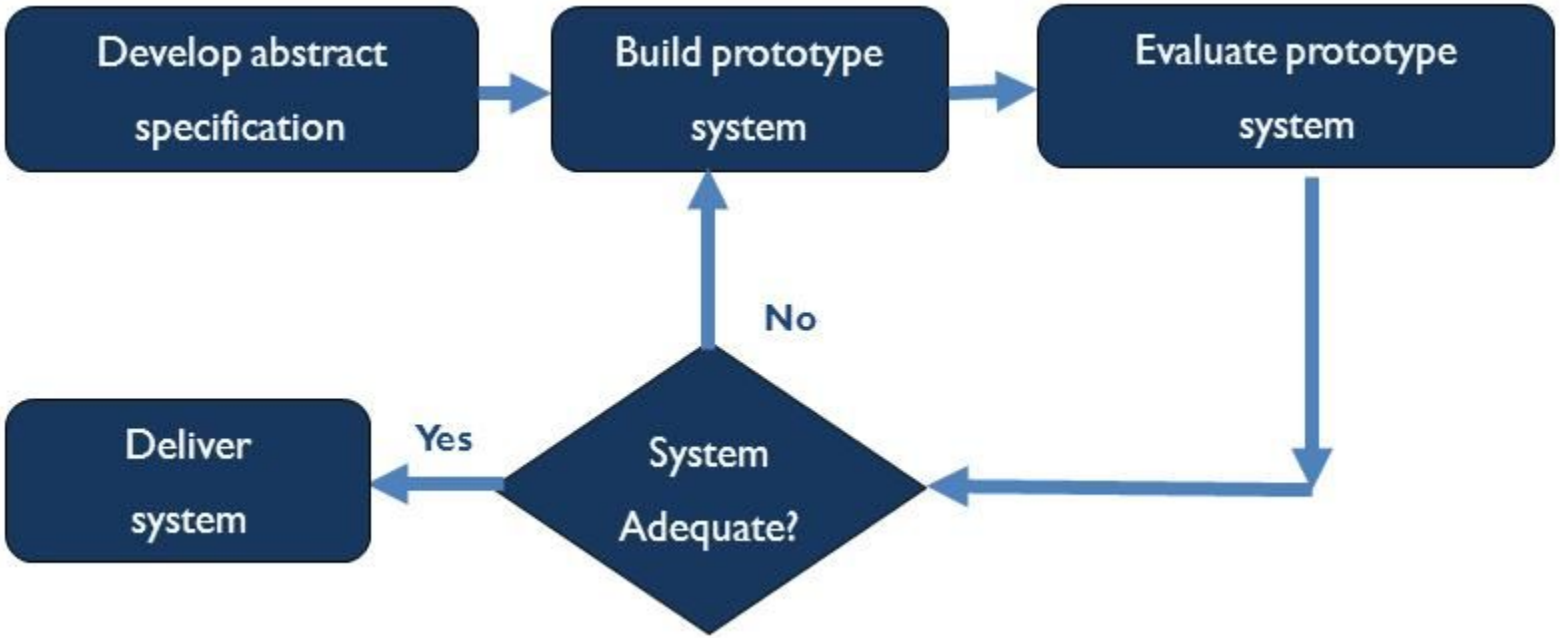
Waterfall - Disadvantage

- Does not allow much reflection and revision;
  - When an application is in testing phase, it’s difficult to go back and modify something that was not well documented;
- No working application is developed until later in the life cycle;
- Difficult to calculate progress within a stage;
- Many risks and uncertainty;

Evolutionary Prototyping

- Initial prototype is first developed, then throughout many phases improvement starts to happen until it reaches the end system;
- Offers the user a working system;
- The system starts with the requirements that are well understood;
- Used for systems that cannot be developed in priori;
- Verification is impossible when there is no specification;

Evolutionary Prototyping - Example



Evolutionary Prototyping - Advantages

- Provides the system in a quick way;
  - Providing the system quickly sometimes is more important than the functionalities details or maintenance easiness;
- Customers commitment with the system;
  - The client's involvement with the system improves the possibility of meeting the requirements;

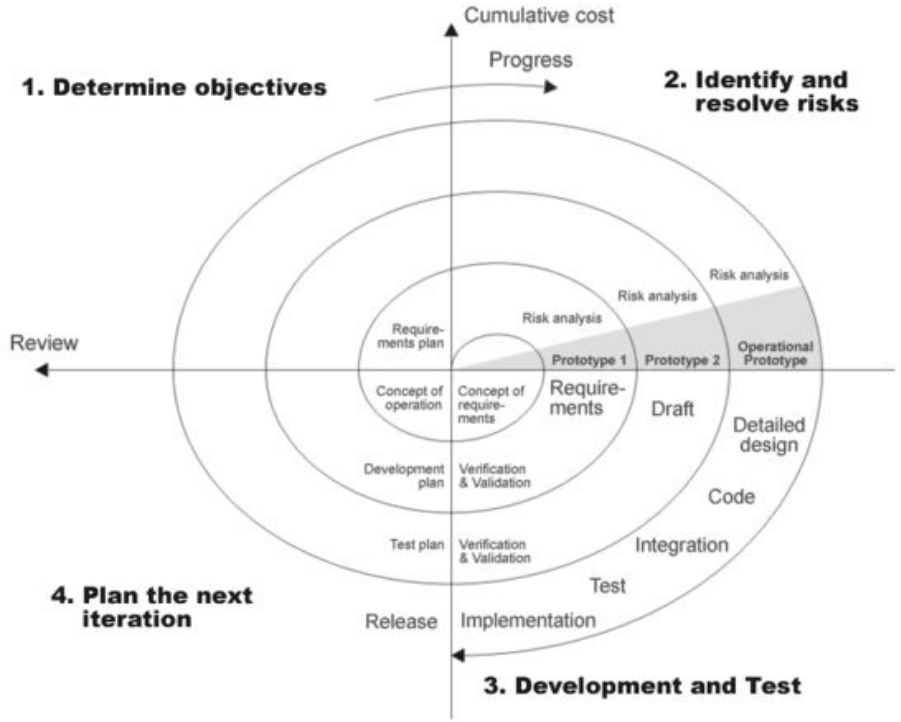
Evolutionary Prototyping - Disadvantages

- The prototype can be discarded;
- Many adjustments can be made on the final prototype;
- The developer can forget inappropriate structures in the prototype;
- Management problems;
  - Costs, documentation;
- Maintenance problems;
  - Technology migration;

Spiral

- Combines iterative development with a systematic control by the Waterfall aspects;
- Developed with a series of evolutionary aspects;
  - The first versions (iterative) can be a model in a paper or a prototype;
  - The last versions (iterative) are more complete of the system;
- The model is divided in a set of activities;
  - Each activity represents a special segment;
  - The activity starts in the center of the spiral in clock-wise;

Spiral - Example



Spiral - Advantages

- Requirement changes can be accommodated;
- Allows extensive use of prototypes;
- Users can see the system earlier;
- Requirements can be found more easily;
- Development can be divided in smaller parts, and the parts with the most risk can be developed before, managing risk in a better;

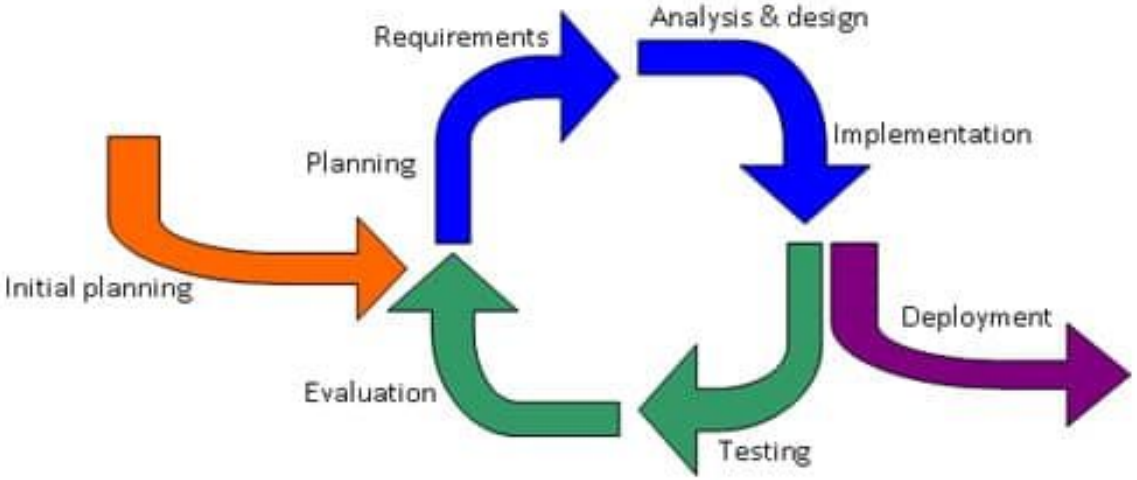
Spiral - Disadvantage

- Management is more complex;
- Due date cannot exist earlier;
- Not appropriate for smaller projects and with small risks;
- Process is complex;
- Spiral can go on forever (without ending);

Iterative Model

- Combines elements from the waterfall model;
- Applied linear sequentiality;
  - Each linear sequence produces “increments” of the Software ready to be delivered;
- The first increment is called “initial planning”;
  - Basic requirements;
- A new plan is developed for the next increment;
  - This process repeats after each new increment be completed;
- Aims to show a working product after each increment;
- Product is defined “complete” when it fulfils all the requirements;

Iterative Development - Example



Model 1: Typical iterative development process

Iterative Development - Advantages

- Some functionalities can be developed fast in the cycle;
- Results can be obtained early and periodically;
- Progress can be measured;
- Low cost when requirements change;
- In each increment, an operational product is delivered;
- Risk analysis is better;
- Good for big and risky projects;
- Tests and debugging is easier;
- Parallel development can be easily planned;

Iterative Development - Disadvantage

- More resources is necessary;
- More management attention is required;
- Not appropriate for smaller projects;
- Requirements definition might need the definition of a full system;
- Due date might not be known;
- Design issues might arise when not all requirements were collected in the beginning of the cycle;

References

1. GHEZZI, Carlo; JAZAYERI, Mehdi; MANDRIOLI, Dino. **Fundamentals of Software Engineering**. 2. ed. New Jersey: Pearson, 2002. 604 p.
2. PRESSMAN, Roger S.; MAXIM, Bruce. **Software Engineering**: A Practitioner's Approach. 8. ed. New York City: Mcgraw-hill Education, 2014. 976 p.