A Conventional Software Process - the Waterfall

Oliver Au
oau@ouhk.edu.hk

Computing, The Open University of Hong Kong

http://ouhk.seprofession.com/

Unit Objectives

After studying this chapter, you should be able to

1. explain that the three driving forces behind a successful software are: people, technology and process
2. describe the well-known software process model: the **waterfall** model with its strengths and weaknesses

Outline

Three Driving Forces for Successful Projects

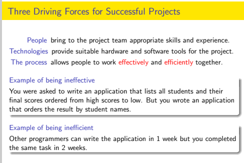
People	bring to the project team appropriate skills and experience.
Technologies	provide suitable hardware and software tools for the project.
The process	allows people to work effectively and efficiently together.

Example of being ineffective

You were asked to write an application that lists all students and their final scores ordered from high scores to low. But you wrote an application that orders the result by student names.

Example of being inefficient

Other programmers can write the application in 1 week but you completed the same task in 2 weeks.


Three Driving Forces for Successful Projects

People bring to the project team appropriate skills and experience.
Technologies provide suitable hardware and software tools for the project.
The process allows people to work effectively and efficiently together.

Example of being ineffective
You were asked to write an application that lists all students and their final scores ordered from high scores to low. But you wrote an application that orders the result by student names.

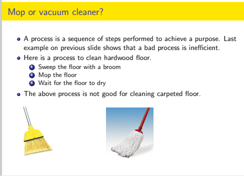
Example of being inefficient
Other programmers can write the application in 1 week but you completed the same task in 2 weeks.

The two terms highlighted in red confuse some people. Effectiveness is about doing the right thing in good quality. It emphasises the end result of your work.

Efficiency is about completing a task without wasting time or resources. It cares about how you do something. In other words, it focuses more on the process using just the required time, effort and materials but no more.

Mop or vacuum cleaner?

- A process is a sequence of steps performed to achieve a purpose. Last example on previous slide shows that a bad process is inefficient.
- Here is a process to clean hardwood floor.
  1. Sweep the floor with a broom
  2. Mop the floor
  3. Wait for the floor to dry
- The above process is not good for cleaning carpeted floor.



Mop or vacuum cleaner?

- A process is a sequence of steps performed to achieve a purpose. Last example on previous slide shows that a bad process is inefficient.
- Here is a process to clean hardwood floor.
  1. Sweep the floor with a broom
  2. Mop the floor
  3. Wait for the floor to dry
- The above process is not good for cleaning carpeted floor.

A broom is on the left. A mop is on the right. The word *mop* can be used as a noun or a verb.

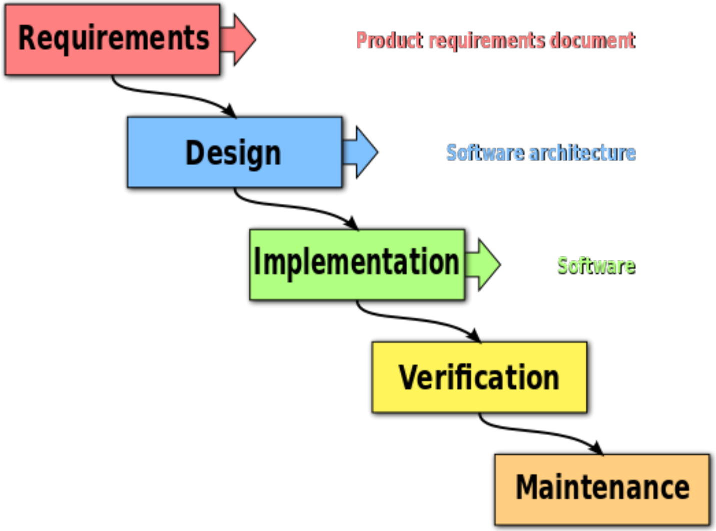
To clean carpeted floor, we need a different process with a different tool. We will use a vacuum cleaner instead of a broom and a mop. Someone good at using brooms and mops may not be good at using a vacuum cleaner.

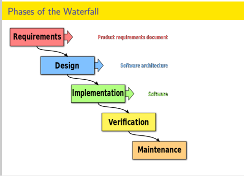
This course teaches you different processes and tools in SE. In your future software development projects, you can adopt the suitable process and tools to suit the requirements.

Choose a model for your project

- Also called **software life cycle model**, a **software process model** captures the essence of a family of similar software processes. .
- The choice of software process model for a project should depend on the purpose and situation of the project.
- If the requirements are well understood and stable, the **waterfall model** may be good.
- If the requirements are unclear or likely to change frequently, the **Scrum model** would be good.

Phases of the Waterfall




Phases of the Waterfall

Waterfall is the best known and likely the most popular software model.

There may be variations in the phase names used, e.g. implementation may be called coding and verification may be called testing.

Some authors may show additional phases. For example a deployment phase may be inserted between verification and maintenance.

Conventionally, the next phase does not begin until the previous has concluded in the Waterfall model.

Project planning, strengths and weaknesses

Planning

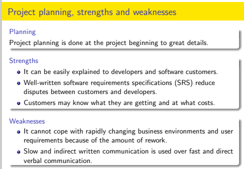
Project planning is done at the project beginning to great details.

Strengths

- It can be easily explained to developers and software customers.
- Well-written software requirements specifications (SRS) reduce disputes between customers and developers.
- Customers may know what they are getting and at what costs.

Weaknesses

- It cannot cope with rapidly changing business environments and user requirements because of the amount of rework.
- Slow and indirect written communication is used over fast and direct verbal communication.



Project planning, strengths and weaknesses

Planning

Project planning is done at the project beginning to great details.

Strengths

- It can be easily explained to developers and software customers.
- Well-written software requirements specifications (SRS) reduce disputes between customers and developers.
- Customers may know what they are getting and at what costs.

Weaknesses

- It cannot cope with rapidly changing business environments and user requirements because of the amount of rework.
- Slow and indirect written communication is used over fast and direct verbal communication.

SRS contains detailed information of what the software product is expected to do. If well-written, it can be used to resolve disputes between customers and developers. SRS can be used to estimate the project costs (though the estimates are seldom accurate).

On the surface, detailed SRS and project plans are great. But changes on the SRSs and project plans are often necessary for various reasons. For example, requirements may change or the project team encounter obstacles. None of these are good for a waterfall project.

Artefacts of the Phases

Requirements Business analysts tries to find out what the customers want and create

- **problem statement** (a short document)
- **SRS** (a long document)
- **project plan** created alongside or after SRS

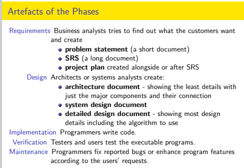
Design Architects or systems analysts create:

- **architecture document** - showing the least details with just the major components and their connection
- **system design document**
- **detailed design document** - showing most design details including the algorithm to use

Implementation Programmers write code.

Verification Testers and users test the executable programs.

Maintenance Programmers fix reported bugs or enhance program features according to the users' requests.



Artefacts of the Phases

Requirements Business analysts tries to find out what the customers want and create
- **problem statement** (a short document)
- **SRS** (a long document)
- **project plan** created alongside or after SRS

Design Architects or systems analysts create:
- **architecture document** - showing the least details with just the major components and their connection
- **system design document**
- **detailed design document** - showing most design details including the algorithm to use

Implementation Programmers write code.

Verification Testers and users test the executable programs.

Maintenance Programmers fix reported bugs or enhance program features according to the users' requests.

An artefact, or in U.S. spelling artifact, is another name for a work product. Each phase produces some artefacts needed by a later phase. For example, the requirements phase produces a requirement specification that enables an architect or analyst to create design documents in the design phase.

Three common design documents are listed here. The architecture document has the least details. System design documents have additional details while detailed design documents have most details.

Before a system is released for production use, it should be tested thoroughly by programmers, testers and users. The key work product of this phase is the fully debugged executable programs. Other work products of this phase are test cases and test logs. Test cases describe the scenarios mimicking the software in actual usage. Test logs document when, by whom and the outcomes of running the individual test suites. A test suite is a collection of tests.

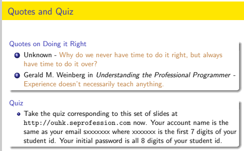
Quotes and Quiz

Quotes on Doing it Right

1. Unknown - Why do we never have time to do it right, but always have time to do it over?
2. Gerald M. Weinberg in *Understanding the Professional Programmer* - Experience doesn't necessarily teach anything.

Quiz

- Take the quiz corresponding to this set of slides at `http://ouhk.seprofession.com` now. Your account name is the same as your email sxxxxxxx where xxxxxxx is the first 7 digits of your student id. Your initial password is all 8 digits of your student id.



Quotes and Quiz

Quotes on Doing it Right

1. Unknown - Why do we never have time to do it right, but always have time to do it over?
2. Gerald M. Weinberg in *Understanding the Professional Programmer* - Experience doesn't necessarily teach anything.

Quiz

- Take the quiz corresponding to this set of slides at `http://ouhk.seprofession.com` now. Your account name is the same as your email sxxxxxxx where xxxxxxx is the first 7 digits of your student id. Your initial password is all 8 digits of your student id.

The first quote warns us about the price we will pay if we do a lousy on our first attempt and have to spend a lot of effort to correct our mistakes. This is certainly true with the lead content found in the water in many buildings. Since they did not do a good job the first time. It is more expensive to fix the mess. Software projects are the same.

Gerald Weinberg is a founding member of a very famous journal the *IEEE Transactions on Software Engineering*. His best known books *Psychology of Computer Programming* and *Introduction to General Systems Thinking*. His quotes says if you do not reflect on your experience, you will not gain from it. In other words, without reflection, you are just wasting your time.