International University, VNU-HCMC

School of Computer Science and Engineering

A Software development in practice

Instructor: Nguyen Thi Thuy Loan

nttloan@hcmiu.edu.vn, nthithuyloan@gmail.com
https://nttloan.wordpress.com/

1

Acknowledgement

- The following slides refers form Cornell University-Computing and Information Science

2

Outlines

- Introduction to practical software development
- Steps in the software development process
- Examples of software development processes

Aim of the Course

We assume that you already know a good deal about computing. can program reasonably, can learn more on the job.

- But success or failure in software development depend on many factors beyond writing good code.
- When you leave IU, you are going to work on production projects where success or failure may cost millions of dollars.

Soon you may be in charge. It may be your money.

- We want you to make your mistakes now and learn from your mistakes.

Academic Integrity & Professional Practice

Software Engineering is a collaborative activity. You are encouraged to work together, but

- Some tasks may require individual work.
- Always give credit to your sources and collaborators.

Good professional practice: To make use of the expertise of others and to build on previous work, with proper attribution.

Unethical and academic plagiarism: To use the efforts of others without attribution.

Variety

Software products are very varied:

- *Applications:* web, smartphone
- *System software:* operating systems, compilers
- *Communications:* routers, telephone switches
- *Data processing:* telephone billing, pensions
- *Real time:* air traffic control
- *Embedded software:* device drivers, controllers
- *Mobile devices:* digital camera, GPS, sensors
- *Information systems:* databases, digital libraries
- *Offices:* word processing, video conferences
- *Scientific:* simulations, weather forecasting
- *Graphical:* film making, design. *etc.,....*

Variety

The craft of software development

Software products are very varied

- Client requirements are very different
- There is no standard process for software engineering
- There is no best language, operating system, platform, database system, development environment, etc.

A skilled software developer knows about a wide variety of approaches, methods, tools. The craft of software development is to select appropriate methods for each project and apply them effectively.

What is Good Software?

General characteristics:

- Functionality work
- Usability
- Maintainability
- Dependability efficiency
- Good software products require good programming.
- Programming quality is the means to the end, not the end itself.

Software is Expensive

- The biggest cost is usually the salaries of the development team

Who is paying the money?

What does that person or organization want?

- What is success?
- What is failure?

Technical people may have very different criteria of success from the people in charge of the organization.

Clients, Customers, and Users

Client

- The client is the person for whom the software development team creates the software.
- The client provides resources and expects some product in return.
- The client is often a member of the organization that is providing the money.
- The client's job success may depend on the success of the software project.

Client satisfaction is a primary measurement of success in a software project.

Clients, Customers, and Users

Customer

- The customer is the person who buys the software or selects it for use by an organization.

User

- A user is a person who actually uses the software.
- With personal software, the user may be the same person as the customer.
- In organizations, the customers and the users are usually different.
- The International Finance Office uses Microsoft Excel
- Who is the customer? Who are the users?

Risk

Many (probably most) software development projects run into difficulties

Problems:

- Does not work as expected (Function)
- Over budget (COST)
- Late delivery (TIME)

Competing goals

Every software project has a trade-off between function, cost, and time.

Extra function adds extra costs for development, testing, maintenance, etc.

What is important to the person who is paying?

Risk

Much of software is wasted (perhaps 50% is never used)
Many software projects fail because the software developers build the wrong software.

The software development team must:

- Understand what the client expects of the software
- Understand what the client's organization expects of the client

The software development team will often:

- Add technical insights and suggestions, but remember:

Client satisfaction is a primary measurement of success in a software project.

Risk

Failures of software development projects can bankrupt companies.
What is the penalty to the client if software is:

- late?
- over budget?
- does not work or full of bugs?

Failures of a software development project will often cost senior executives their jobs.

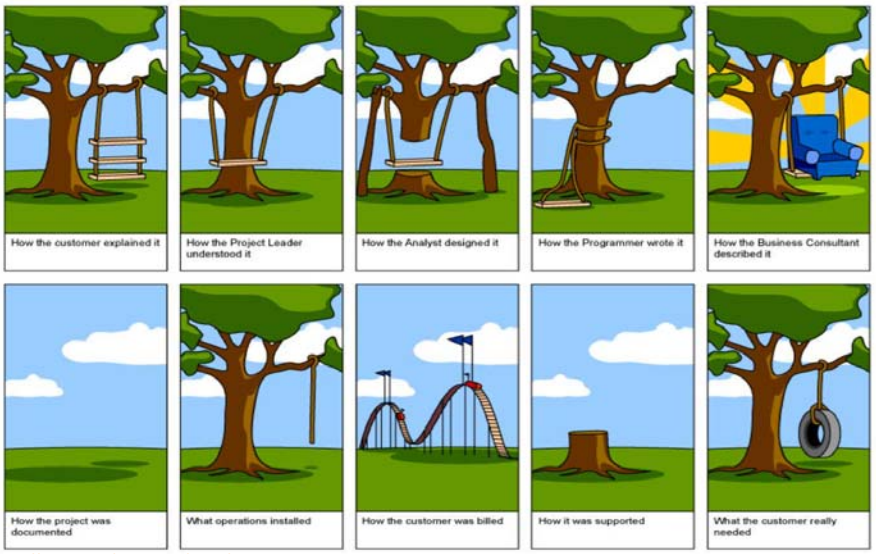
Example: Apple’s mapping app

Is Software Development Risk Costing You Money?

- Poor software project management often means missed deadlines, cost overruns or even outright failure of the project.
- How can your company avoid this industry-wide problem?

https://www.projectsmart.co.uk/is-software-development-risk-costing-you-money.php

Example: The Software Design & Development Process Tree-Swing



https://twitter.com/soluppgang/status/624475066076647424

Minimizing Risk

Communication with the Client

- Feasibility studies (whether to begin a project).
- Separation of requirements (what the client wants) from design (how the developers meet the requirements).
- Milestones (how the developers report or demonstrate progress to the clients) and releases.
- Acceptance (how the client tests that the software meets the requirements) and user testing.
- Handover (ensuring that the client receives a package that can be operated and maintained over a long time period).

Minimizing Risk

Visibility:

The people who take the responsibility must know what is happening

The problem (as seen by a manager)

Must rely on others for reports of progress or difficulties

... *but s*oftware developers

- Have difficulty evaluating progress
- Are usually optimistic about progress
- Consider reporting a waste time, *etc.*

Working software provides excellent visibility.

You will make regular progress reports on your projects

Minimizing Risk

Short Development Cycles

Short development cycles with frequent releases provide an important methodology that is used by a number of successful companies.

Risk is minimized by frequent delivery of working software (weeks rather than months).

- Client, customers, and users can evaluate the developers' work.
- Opportunities to adapt to changing circumstances.

This is one of the basic principles of agile software development.

Scale

Large and very large systems have different needs

- Every large system is developed by many people, who are constantly changing.
- Before a big project is completed the requirements have changed many times.
- No large system is ever complete.
- Nobody comprehends more than a fraction of the project.

Teams

Most software development is by teams.

- Effectiveness of team determines success.

Most large software projects are built on older ones.

- It is rare to start a new system from scratch.
- Building on the work of others is a fundamental skill of software development.
- Much software is built in increments, with different teams responsible for the increments.

Managing Large Projects

Large software projects need skilled management

- Project management
- Personnel management
- Economic, legal, and social environment

Development processes

- Iterative refinement
- Spiral development
- Agile
- Sequential

Software Development Processes

Fundamental Assumption:

- Good processes lead to good software
- Good processes reduce risk
- Good processes enhance visibility
- Good processes enable teamwork

Variety of Software Processes

Software products are very varied...

- Therefore, there is no standard process for all software engineering projects
- BUT successful software development projects all need to address similar issues.
- This creates a number of process steps that should be part of all software projects.

Basic Steps in all Software Development

Feasibility and planning

Requirements
User interface design
System design
Program development (design and coding)
Acceptance and release

These steps may be repeated many times during the development cycle

Operation and maintenance

It is essential to distinguish among these steps and to be clear which you are doing at any given moment.

Note

- Considerations of testing, security and performance are part of many of these steps.

Feasibility

A feasibility study precedes the decision to begin a project.

- What is the scope of the proposed project?
- Is the project technically feasible?
- What are the projected benefits?
- What are the costs, timetable?
- Are the resources available?
- What are the risks and how can they be managed?

A feasibility study leads to a decision: go or no-go.

Requirements

- Requirements define the function of the system from the client's viewpoint.
- The requirements establish the system's functionality, constraints, and goals by consultation with the client, customers, and users.
- They must be developed in a manner that is understandable by both the client and the development staff.

Requirements

This step is sometimes divided into:

- Requirements analysis
- Requirements definition
- Requirements specification

The requirements may be developed in a self-contained study, or may emerge incrementally.

Failure to agree on the requirements and define them adequately is one of the biggest cause of software projects failing.

User Interface Design

Usability is of great importance in many modern applications and software systems. That requires good user interface design.

User interfaces need to be evaluated with users.

This requires iterative development:

- Design the user interface
- Test with users
- Revise the user interface
- Repeat

System and Program Design

Design describes the system from the software developers' viewpoint.

System design:

- Establish a system architecture, both hardware and software, that matches the requirements
- Security and performance are parts of the system design.

System and Program Design

Program design:

- Represent the software functions in a form that can be transformed into one or more executable programs
- Preliminary user testing is often carried out as part of the design step.
- Models are used to represent the requirements, system architecture, and program design. This course teaches the basic concepts of the Unified Modeling Language (UML).

Implementation

Implementation (coding)

- The software design is realized as a set of programs or program units.
- These software components may be written by the development team, acquired from elsewhere, or modified from existing components.

Program testing

- Program testing by the development staff is an integral part of implementation.
- Individual components are tested against the design.
- The components are integrated and tested against the design as a complete system.

Acceptance and Release

Acceptance testing

- The system is tested against the requirements by the client, often with selected customers and users.

Delivery and release

- After successful acceptance testing, the system is delivered to the client and released into production or marketed to customers.

Operation and Maintenance

Operation:

The system is put into practical use.

Maintenance:

Errors and problems are identified and fixed.

Evolution:

The system evolves over time as requirements change, to add new functions, or adapt to a changing technical environment.

Phase out:

The system is withdrawn from service.

This is sometimes called the Software Life Cycle

Quality Control in all Software Development

- Validating the requirements
- Validating the system and program design

- Usability testing
- Program testing
- Acceptance testing

- Bug fixing and maintenance

Some of these steps will be repeated many times during the life of the system.

Categories of Testing

The term “testing” is used in several different contexts, which are easily confused:

User testing

Versions of the user interface are tested by users. Their experience may lead to changes in the requirements or the design.

Program testing

The development team tests components individually (unit testing) or in combination (system testing) against the design to find bugs, etc.

Acceptance testing

The client tests the final version of the system or parts of the system against the requirements.

Sequence of Processes

Every software project will include these basic steps, in some shape or form, but:

- The steps may be formal or informal
- The steps may be carried out in various sequences

Software Development Processes

Major alternatives

In this course, we will look at four categories of software development processes:

- **Iterative refinement:**

  Go quickly through all the steps to create a rough system, then repeat them to improve the system.

- **Spiral:**

  A variant of iterative refinement in which new and updated components are added to the developing system as they are completed.

Software Development Processes

- **Agile development:**

  Small increments of software are developed in a sequence of sprints, each of which creates deployable code.

- **Waterfall model:**

  Complete each process step before beginning the next.

Heavyweight and Lightweight (HaL) Software Development

In a **heavyweight process**, the development team works through the steps slowly and systematically, with the aim of fully completing each step and delivering a complete software product that will need minimal changes and revision.

Example: Modified Waterfall Model

In a **lightweight process**, the development team releases working software in small increments, and develops the plans incrementally, based on experience. Each increment includes all the process steps. There is an expectation that changes will be made based on experience.

Example: Agile Software Development

HaL Methodologies

Heavyweight	⟷	Lightweight
Processes and tools	⟷	Individuals & interactions
Specification	⟷	Working software
Client negotiation	⟷	Client collaboration
Following a plan	⟷	Responding to change

Based on the Manifesto for Agile Software Development:
http://agilemanifesto.org/

History

Software engineering, as a discipline, dates from the early 1970s.

At that time:

- Most computer systems were conversions of systems that had previously been done manually, e.g., payroll, billing, airline reservations. The requirements were well understood.
- Many systems followed the same architecture, Master File Update. The system design was well understood.
- Coding was tedious with none of the modern languages and tools. Therefore, it was important to have a good program design before beginning to code.

These factors led to the Waterfall Model of software development.

Definitions: Activity and Sprint

Activity

An activity is a general term for any part of a project that takes place over time (also known as a task)

- Each step in the software development process can be broken down into several activities.

Sprint

A sprint is a set period of time during which a team completes part of a software project.

- Each sprint will go through most or all of the process steps.
- A typical sprint might have a team of 6 to 8 people working for 2 to 4 weeks.

Iterative Refinement

Concept

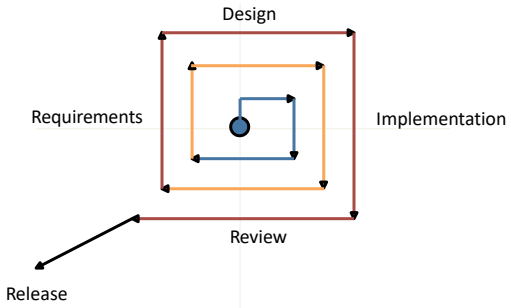
- Create a prototype system early in the development process.
- Review the prototype with clients and test it with users, to improve the understanding of the requirements and clarify the design.
- Refine the prototype in a series of iterations.

Requirements are hard to understand until there is an operational system, particularly with user interfaces.

Mistakes in the requirements are the most expensive to correct.

Example: Converting a national archive from paper based to computer based.

Iterative Refinement



Discussion of Iterative Refinement

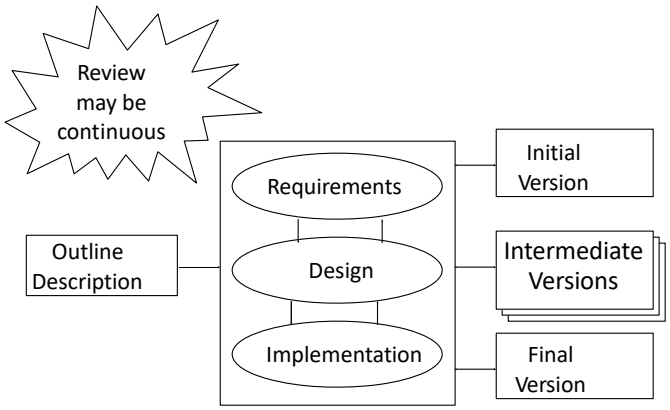
This is a medium weight process with documentation created during the process.

Iterative refinement uses various techniques that enable the client to review the the planned system early during development:

- User interface mock-up
- Throw-away software components
- Dummy modules
- Rapid prototyping
- Successive refinement

Get something working as quickly as possible, for client and user evaluation, but do not release it.

Iterative Refinement with a Large System



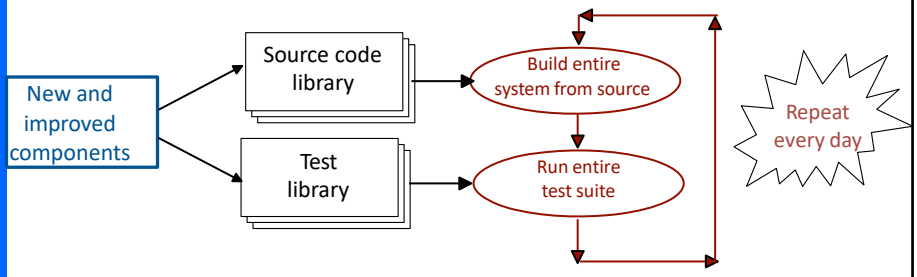
Spiral Development

- Create a base system that has the overall structure of the final product with dummy stubs for missing components.
- Create a comprehensive set of test cases for all completed components
- Use a succession of sprints to develop new or improved components, each with a set of test cases. Add these components to the source code library.
- On a daily cycle, build the entire system from the source code library and run the complete set of test cases.

With spiral development there is always a fully tested system, but the functionality is incomplete.

Example: Developing a new version of an operating system.

Spiral Development



Discussion of Spiral Development

Spiral development is widely used to develop new versions of large systems

- The overall system architecture is well understood.
- Large components can be developed and tested separately.
- The importance of the system justifies the overhead of setting up a comprehensive set of automated tests.

Challenges

- Difficult to make major changes to the architecture.
- Difficult to make changes that effect many components.

Incremental Release of Online Systems

It is easier for a small team to develop a small system correctly than to coordinate large projects with many ramifications.

With web sites and other online software it is often possible to release a very basic system and add extra functionality in a sequence of short sprints.

Example:

- Start-up company developing a web based shopping service.

Incremental Release of Online Systems

Advantages

- Pay-back on investment begins soon.
- Requirement are more clearly understood in developing subsequent sprints – minimize waste.
- Feedback from customers and clients can be incorporated in later phases.

It is easier for a small team to develop a small sprint correctly than to coordinate large projects with many ramifications.

Agile methods

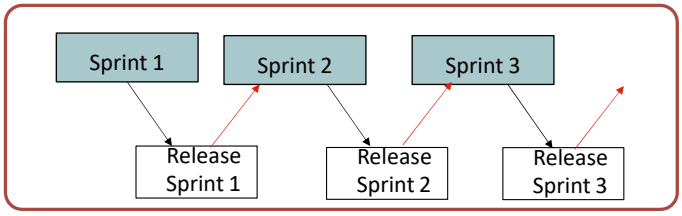
The term Agile is used for a variety of methods.

General principles:

- A large projects is divided into small increments called sprints. The development is carries out by all teams of 4 to 9 people.
- The schedule is divided into fixed time boxes, perhaps 2 to 4 weeks.
- Each sprint is a time box during which the team completes part of a software project. A single sprint will go through several process steps, such as requirements, design, coding, and testing.
- Each sprint ends with fully tested code, ready to be put into production.

This is a lightweight process with minimal documentation created during the process, but the final version needs full documentation for future maintenance

Agile Development



After each sprint the code may be:

- released (original agile method)
- combined with code from other sprints for subsequent release
- incorporated into a larger code base (spiral development)

Agile: Releasing code

Versions of agile

- The original version of agile required each sprint to end with released code.
  This is rarely possible in practice.
- In this course, we define a sprint to create production quality code.
- Some people use the term “sprint” to cover any short activity, but this is beyond the agile spirit.

Agile development: Rework

The challenge of agile development

- The agile approach is excellent for the development or continual enhancement of a system within an established architecture.
- A high-level team must establish the overall architecture and coordinate the sprints.

Agile development: Rework

Rework

With agile development the requirements and design of the overall system emerge incrementally.

- Inevitably parts of some early sprints will need to be reworked.
- This requires changes to code that has already been fully tested and may have been released. This is always awkward.

If the volume of rework is large, it is more efficient not to fully polish each component, but to use iterative refinement to minimize the amount of rework.

Discussion of Agile Development

Variations on agile software development

In practice it is rarely possible for every sprint to end with released software, but software development based on sprints has many advantages.

Modern software development includes a wide range of processes that are called “agile”. Other processes with names such as “heroic programming” or “scrum” use many of the same concepts.

Discussion of Agile Development

Characteristics

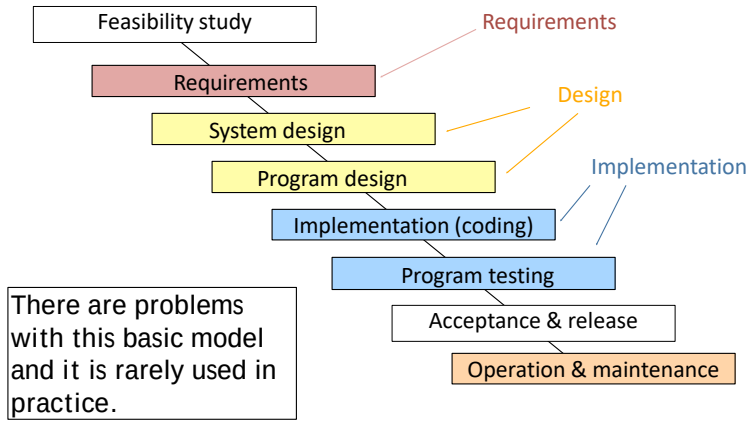
Development of a project is divided into a large number of sprints. Each sprint ends with fully tested code.

This is a lightweight process with minimal documentation created during the process.

A general definition (from Wikipedia)

Agile software development describes a set of principles for software development under which requirements and solutions evolve through the collaborative effort of self-organizing cross-functional teams.

Sequential Development: The Waterfall Model



There are problems with this basic model and it is rarely used in practice.

Discussion of the Waterfall Model

The waterfall model is a heavyweight process with full documentation of each process step.

Advantages:

- Process visibility
- Separation of tasks
- Quality control at each step
- Cost monitoring at each step

Disadvantages:

In practice, each stage in the process reveals new understanding of the previous stages, which often requires the earlier stages to be revised.

The Waterfall Model is not flexible enough.

Discussion of the Waterfall Model

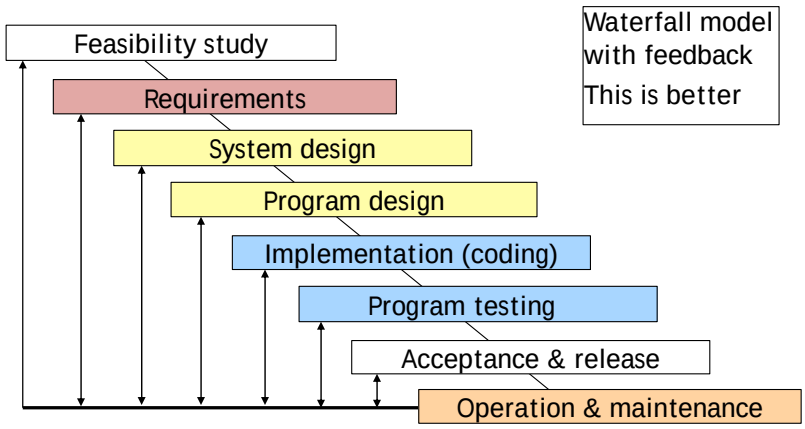
A pure sequential model is impossible

Examples:

- A feasibility study cannot create a proposed budget and schedule without a preliminary study of the requirements and a tentative design.
- Detailed design and implementation reveal gaps in the requirements specification.
- Requirements and/or technology may change during the development.

The plan must allow for some form of iteration.

Modified Waterfall Model



Sequential Development

Sequential processes work best when the requirements are well understood and the design is straightforward, e.g.,

- Conversions of manual data processing systems where the requirements were well understood and few changes were made during the development (e.g., electricity billing).
- New models of a product where the functionality is closely derived from an earlier product (e.g. automatic braking system for a car).
- Portions of a large system where some components have clearly defined requirements and are clearly separated from the rest of the system.

Contracts

Note about contracts for software development

Some organizations contract for software development by placing separate contracts for each stage of the Waterfall Model or arrange for payment after each stage. This is a very bad practice.

Mixed Processes

In practice, many large projects use processes that mix aspects of the four types of software process.

For example:

- With spiral development, new components may be developed using any of the three other methods.
- User interfaces have to be tested with users. This forces iterative development, even within an agile or sequential process.

Mixed Processes: Phased Development

Combine sequential and iterative elements

A simple system with basic functionality is brought quickly into production (Phase 1). This system may be developed using a sequential or iterative refinement.

Subsequent phases are based on experience gained from users of the previous phase.

Advantages

- Pay-back on investment begins soon.
- Requirement are more clearly understood when developing subsequent phases

Examples of Mixed Processes

Iterative Refinement + Waterfall Model

Problem: Add graphics package to a programming environment

Phase 1: Iterative refinement

Make several prototype versions by extending the current environment with a preprocessor and run-time support package. Test with users until users are pleased with function. Throw the code away.

Phase 2: Modified waterfall

Use the results of Phase 1 to specify a formal set of requirements. Write new compiler and run-time system incorporating graphics elements. Make minor adjustments to requirements as needed.

Corporate Processes

Large software development organizations have their own internal processes that are designed for their needs.
For example:

- Amazon.com (Internet commerce) makes extensive use of sprints. Most software development is divided into increments of about four weeks elapsed time.
- Lockheed Martin (government contractor) follows a sequential process that fits with the way that the US government manages software contracts.

Corporate Processes

- SAP (business software) emphasizes the functionality that is seen by their business customers. Much of the development is suitable for a sequential process.
- Microsoft (PC software) places great emphasis on testing with a very wide variety of equipment and backward compatibility. Much of the development uses a spiral process.

Choosing a Software Process

Changes during the software development process are expensive.

- If the requirements are poorly understood, or expected to change, select a process that keeps flexibility. Iterative refinement, agile sprints, phased implementation.
- If a big software system has many inter-related components, avoid major changes to the design of a system during development. Sequential process, such as the modified waterfall model.
- If the market for the software is poorly understood, use a process that gets operational software in front of customers as quickly as possible. Incremental, agile sprints.

Observations about Software Processes

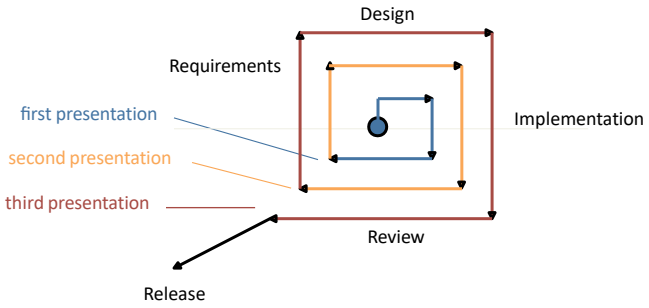
Completed projects should have included all the basic process steps *but* ... the development process is always partly evolutionary.

Risk is lowered by:

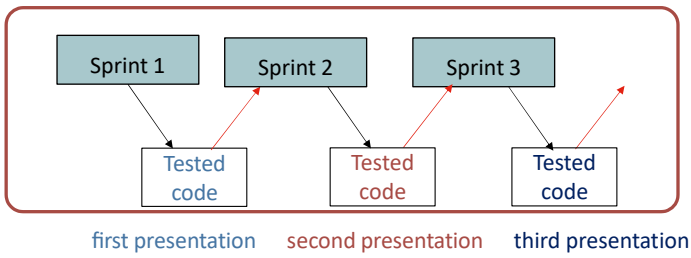
- Prototyping key components
- Frequent releases, or dividing large projects into phases
- Early and repeated testing with users and customers
- Following a visible software process
- Making use of reusable components

It is never possible to complete each step without provision for revision. This is known as throwing it over the wall.

SE Projects: Iterative Refinement

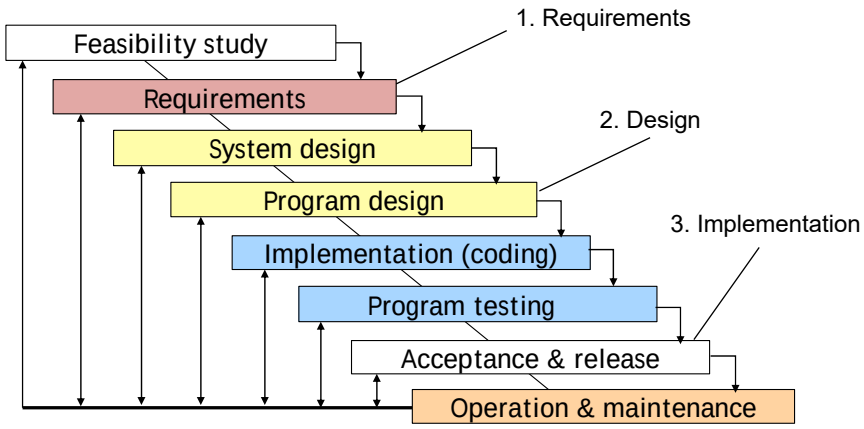


SE Project: Agile Development



For each sprint aim to complete a section of the code.

SE Projects: Sequential Development



If you follow a sequential process the three presentations should be as shown.

Thank you for your attention!