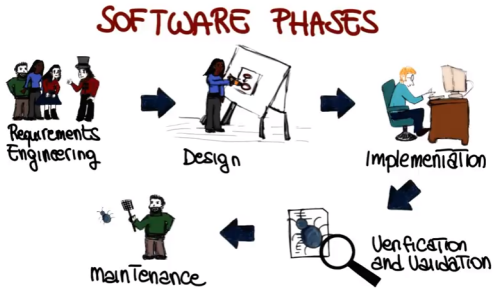
DA-IICT

IT 314: Software Engineering

Software Process and Process Models



Software

Q : If you have to write a 10,000 line program in C or Java to solve a problem, how long will it take?

Answers: generally range from 2-4 months

Let us analyze the productivity

- Productivity = output/input resources
- In SW output is considered as LOC
- Input resources is effort - person months; overhead cost modeled in rate for person month
- Though not perfect, some productivity measure is needed, as project has to keep it high

Software…

The productivity is 2.5-5 KLOC/PM

Q: What is the productivity in a typical commercial SW organization ?

A: Between 100 to 1000 LOC/PM

Q: Why is it low, when your productivity is so high? (people like you work in the industry)

A: What the student is building and what the industry builds are two different things.

What is the difference between a student program and industrial strength SW for the same problem?

Software in 20's

Much Better

Better processes, practices, and tools are being used

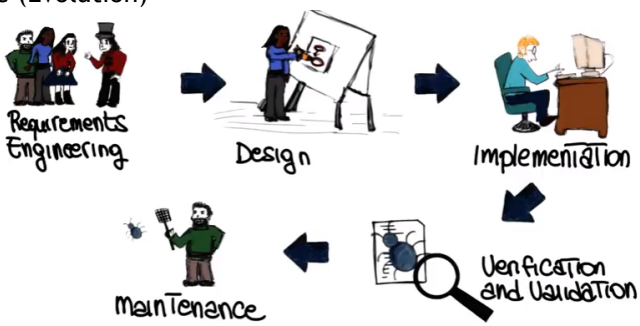
Failure rate is dropped to 30-40%

Possible due to the advances in the field of "Software Engineering"

Software Development Life Cycle

Consists of

- Requirement gathering, Analysis and Specification
- Design (Both high and low level design)
- Construction
- Testing
- Maintenance (Evolution)
- Closure



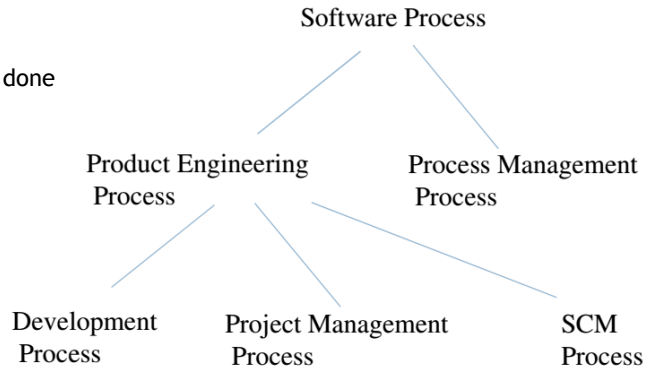
A Dual Emphasis

- Product / Project

what is done

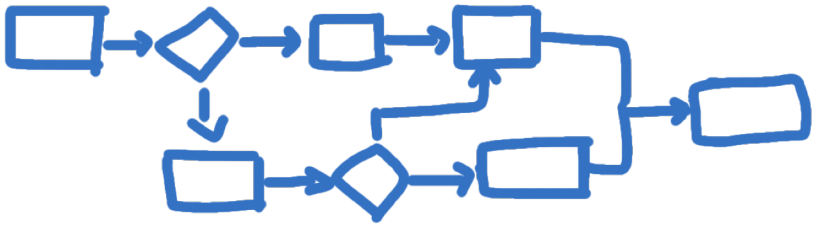
- Process

how things are done



What is a Process

A series of steps involving activities, constraints, and resources that produce an intended outcome of some kind



How a Software Process Help US?

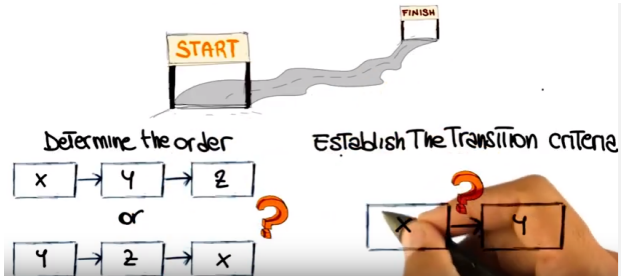
- Imposes consistency and structure on a set of activities.
- Guides our actions by allowing us to examine, understand, control and improve the activities that comprise the process.
- Captures our experience and pass them along to others.

Software Process Characteristics

- Visibility
- Predictability
- Testability
- Maintainability
- Early defect Removal
- Support Change
- Customizability

Software Process Models

- A (software/system) process model is a description of the sequence of activities carried out in an SE project, and the relative order of these activities.
- Determine the order
- Establish the transition criteria



Software Process Models

Also termed as Software Life Cycle Models

•Water-Fall model

•Prototyping model
- Throw away prototyping
- Evolutionary prototyping

•Incremental model (Evolutionary model)
- Incremental Water-Fall model
- Time Boxing model
- Spiral model
- Synchronize & Stabilize model

•Commercially projected models like RUP, XP, SCRUM

Software Process Models

By changing the process model, we can improve and/or tradeoff:

- Development speed (time to market)
- Product quality
- Project visibility
- Administrative overhead
- Risk exposure
- Customer relations, etc, etc.

Normally, a process model covers the entire lifetime of a product.

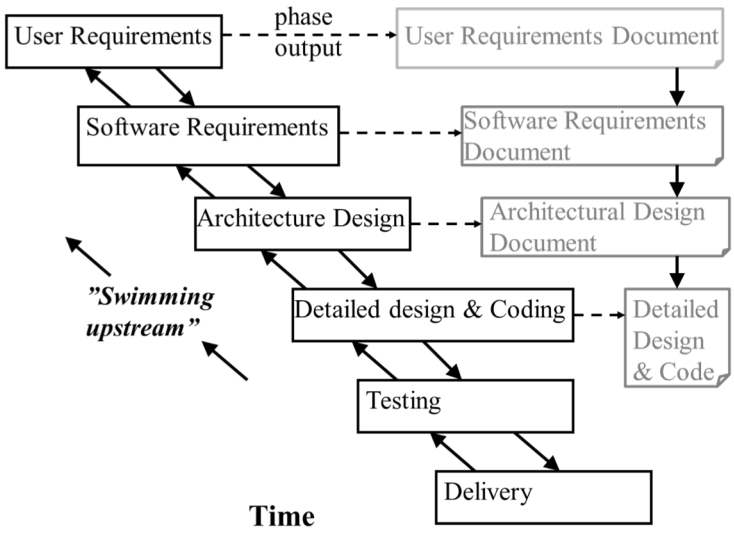
From birth of a commercial idea to final de-installation of last release

i.e. The three main phases: design, build, maintain (50% of IT activity goes here!)

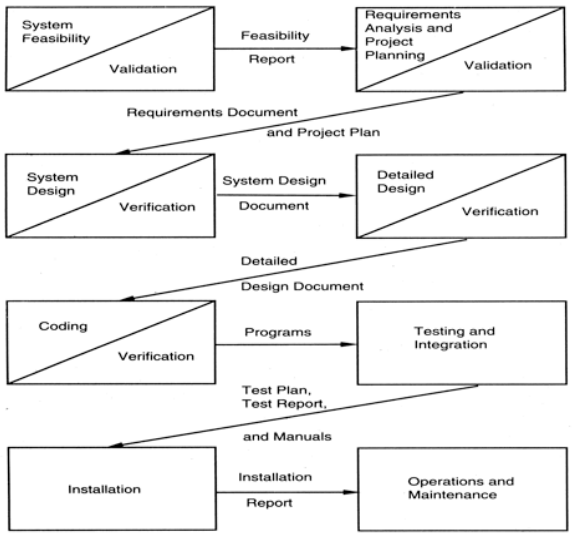
Waterfall Model

- One of the first process development models proposed
- Works for well understood problems with minimal or no changes in the requirements
- Simple and easy to explain to customers
- It presents
  - a very high-level view of the development process
  - sequence of process activities
- Each major phase is marked by milestones and deliverables (artifacts)

Waterfall Model



Waterfall Model



Waterfall...

Advantages

- Simple
- Complete control on the software process
- Better quality control
- Fits other engineering process models: civil, mech etc.
- Reinforces good habits: define-before-design, design-before-code

Disadvantages

- Complete and consistent set of requirements?
- Do not support iteration
- Change management is difficult
- Customer to have patience
- Does not facilitate good use of resources
- May choose outdated technology

Suitable for Well understood problems; Short duration projects; Automation of existing manual systems

Waterfall...

- Waterfall model was used to develop enterprise applications like
  - Customer Relationship Management (CRM) systems,
  - Human Resource Management Systems (HRMS),
  - Supply Chain Management Systems,
  - Inventory Management Systems,
  - Point of Sales (POS) systems for Retail chains etc.

Prototyping

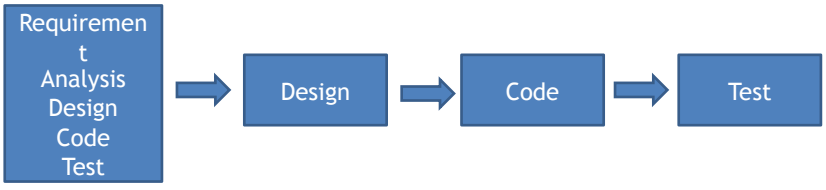
Limitations of Waterfall:

1.Requirements of a system can be frozen

2.Freezing the requirements usually requires choosing the hardware (because it forms a part of the requirements specification)

Prototyping:

Instead of freezing the requirements before any design or code, a prototyping help to understand the requirements.



Prototyping...

Evolutionary Prototyping

- The task of requirement elicitation is easier (+)
- Progress is visible and client is happy (+)
- Better use of resources (+)
- More flexibility compared to water-fall model (+)
- May suffer from “short-sightedness” (-)
- May not estimate RISK correctly (-)
- Disallows later changes (-)

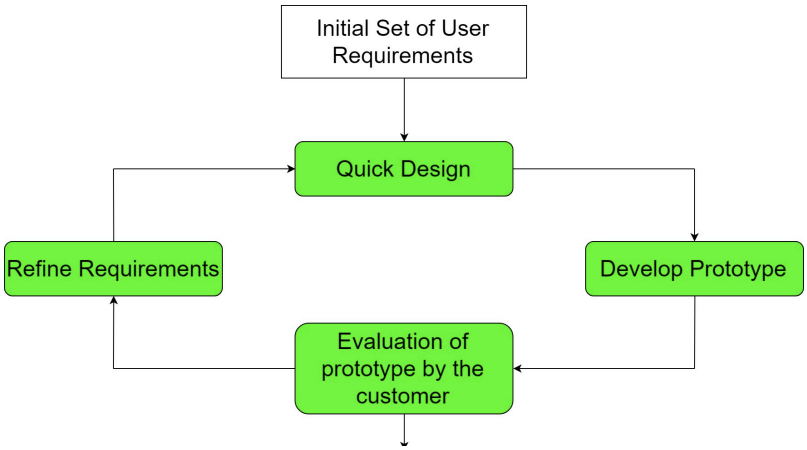
Prototyping...

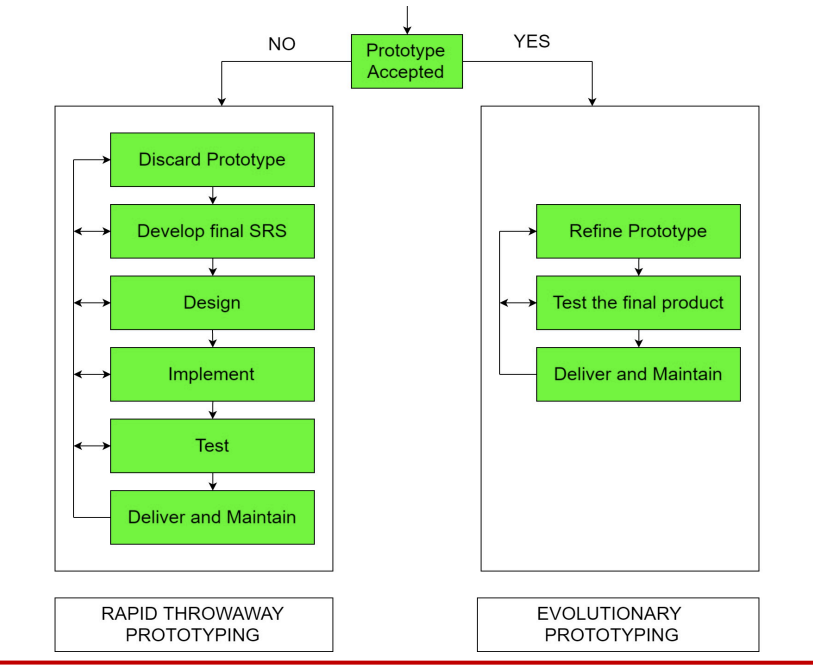
Throw-Away Prototyping

- The task of requirement elicitation is easier (+)
- Progress is visible and client is happy (+)
- Better use of resources (+)
- More flexibility compared to water-fall model (+)
- May improve from “short-sightedness” (+)
- May not estimate RISK correctly (-)
- Disallows later changes (-)

Difference between Throw-Away and Evolutionary Prototyping??

Prototyping...





Prototyping...

The difference is whether you build on the prototype, or whether you discard it when you have completed.

An evolutionary prototype is one that is built such that it can be expanded upon and revised, but does not have to be discarded and completely rewritten in order to go to market.

A throw-away prototype is something that's designed to capture the "essence" of whatever it is that you're prototyping, but that will be completely replaced by something else that will be what goes to market.

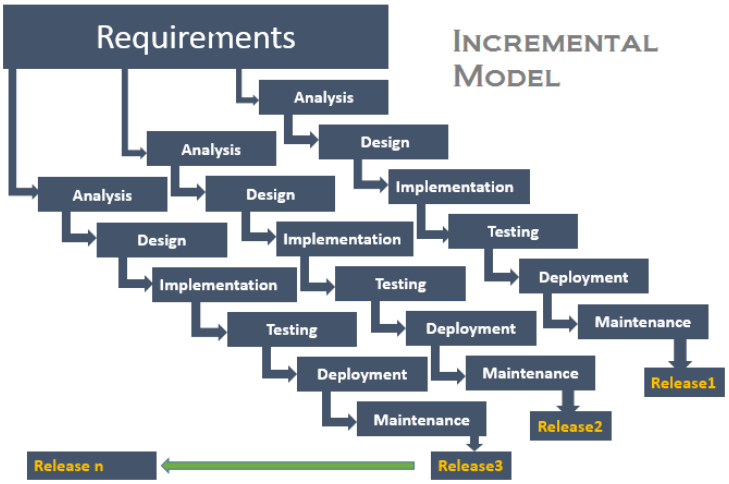
Prototyping Model

Suitable for

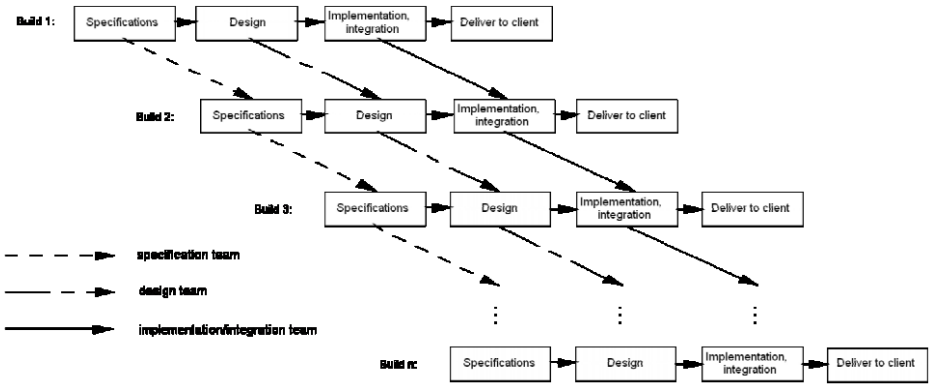
- Less experience teams
- System with novice users
- Requirements are not clear
- UI is very important

Incremental Model

How Evolutionary prototype model evolve?



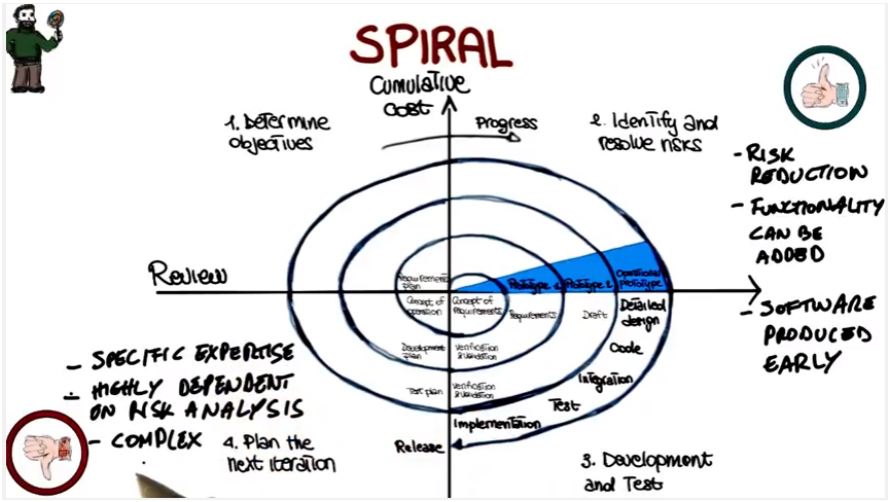
Incremental Waterfall Model



Spiral Model: Incremental model

- Always some risk involved in software development
  - people leave… other products not delivered on time…
- Key idea
  - minimize risk
    - e.g., building prototypes & simulations minimizes risks
- Precede each phase by
  - looking at alternatives
  - risk analysis
- Follow each phase by
  - evaluation
  - planning of next phase

Spiral Model: Combination (Iterative + Waterfall)



Spiral Model

Advantages

- Realism: the model accurately reflects the iterative nature of software development on projects with unclear requirements
- Flexible: incoporates the advantages of the waterfal and rapid prototyping methods
- Comprehensive model decreases risk
- Good project visibility.

Disadvantages

- Needs technical expertise in risk analysis to really work
- Model is poorly understood by non-technical management, hence not so widely used
- Complicated model, needs competent professional management.
- High administrative overhead.

Synchronize & Stabilize Model (Microsoft)

Requirements analysis
- interview potential customers

Draw up overall product specifications

Divide project into 3 or 4 builds
- each adds new functionality (1st gives base)
- each build carried out by small parallel teams

At the end of day – synchronize (test & debug)

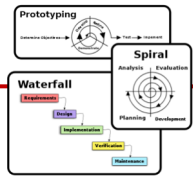
At end of build – stabilize (fix & freeze build)

Incremental Model

Suitable for

- Risk is high
- Requirements are not clear but will evolve
- "Time to Market" is critical
- Big projects
- New concept/product development

Comparison of Process Models



Strengths	Weakness	Type of Projects
Water fall Simple Easy to execute Intuitive and logical	All or nothing approach Requirements frozen early Cycle time too long User feedback not allowed	For well understood problems, short duration project, automation of existing manual systems
Prototyping Helps in requirements Reduces Risk Leads to a better system	Front heavy process Possibly higher cost Disallow later changes	System with novice users When uncertainties in requirements When UI is very important
Iterative/Incremental Regular/quick deliveries Reduces risk Accommodate changes Allow users feedback Allows reasonable exit points Prioritizing requirements	Each planning can have planning overhead Cost may increase as work done in one iteration may have to be undone later	For business where time is of essence Where risk of a long project cannot be taken Where requirements are not known and will be known only with the time

Questions?

Giving reasons for your answer based on the type of system being developed, suggest the most appropriate generic software process model that might be used as a basis for managing the development of the following system.

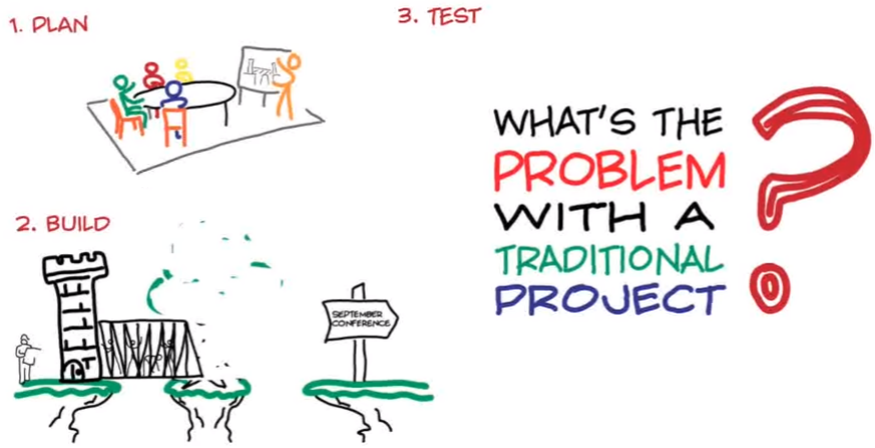
1. a system to control anti-lock braking in a car.
2. a virtual reality system to support software maintenance
3. a university accounting system that replaces an existing system
4. an interactive system that allows railway passenger to find train times from terminals installed in stations

Agile Software Development

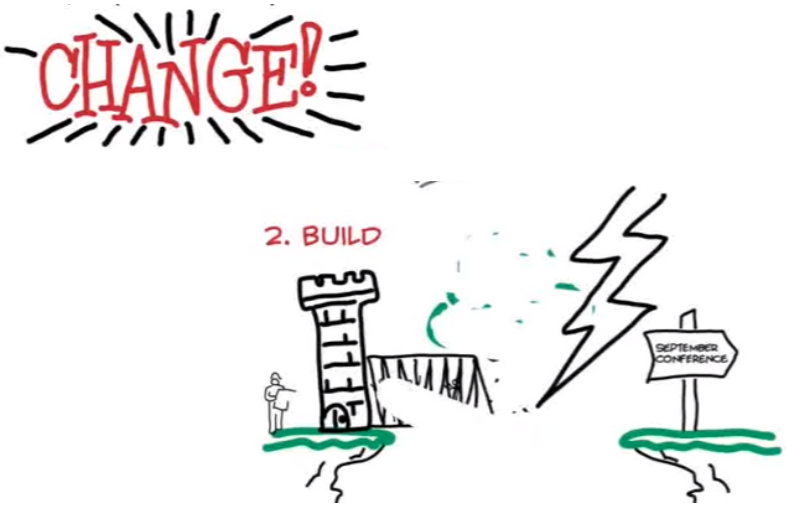
Agile reduces the risk by delivering the value of the project very early

Agile Software Development

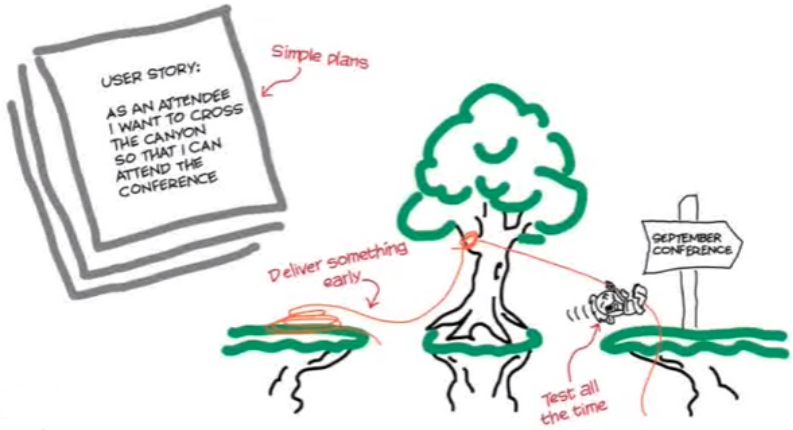
Traditional Software Development – Opposed to Agile



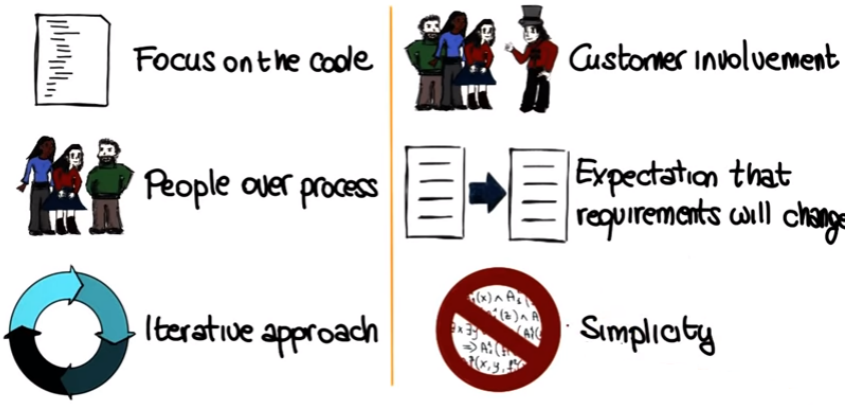
Agile Software Development



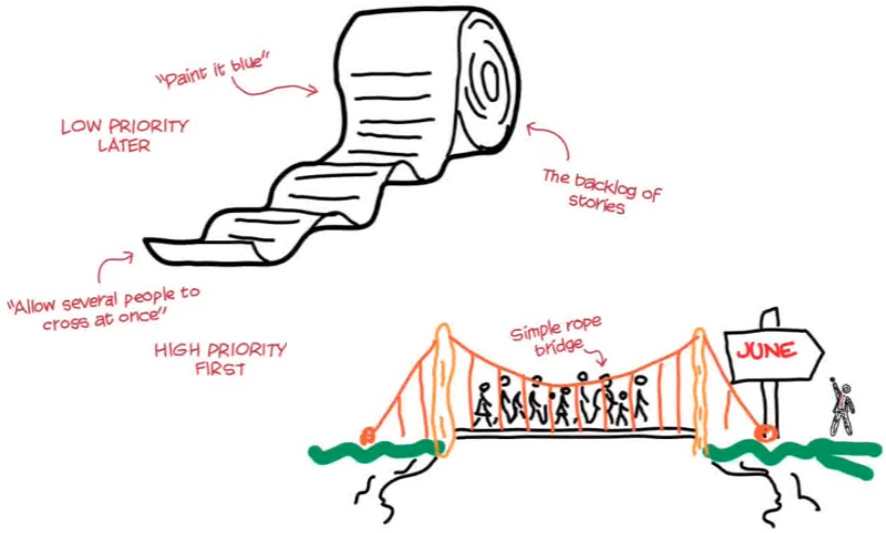
Agile Software Development



Agile Software Development



Managing Requirements - Agile



Questions??

Next Lectures…
Other Process Models
(Time boxing, RUP, Agile (XP, SCRUM) and so on…., CMM