Software Processes

CSC 4700 Software Engineering

Lecture 5

Software Development Processes

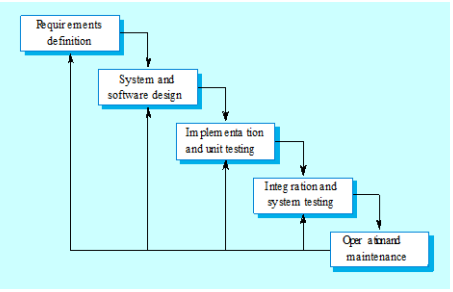
The software process

- A structured set of activities required to develop a software system
  - Specification;
  - Design;
  - Validation;
  - Evolution.
- A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.

The old way

- The way software is engineered has evolved over the years
- The "new" ways of software engineering resemble the "old" ways in a lot of ways
- See if you can make out the resemblance

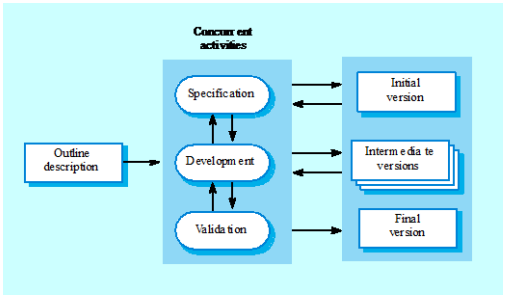
Waterfall model



Waterfall Model

- Inflexible partitioning into distinct stages makes it hard to deal with changing customer requirements.
- Only works when requirements are well-known and few changes are expected… which is rare!
- The waterfall model is still used for some large, multi-site projects, but used less and less

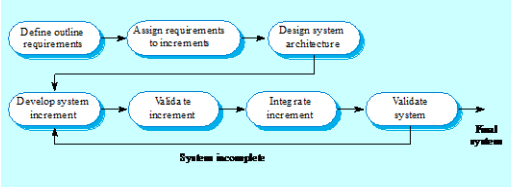
Evolutionary development



Evolutionary development

- Problems
  - Throw-away prototyping might waste work
  - Lack of process visibility
  - Systems are often poorly structured, evolve
  - Special skills (e.g. in languages for rapid prototyping) may be required
- Applicability
  - For small or medium-size interactive systems;
  - For parts of large systems (e.g. the user interface);
  - For short-lifetime systems.

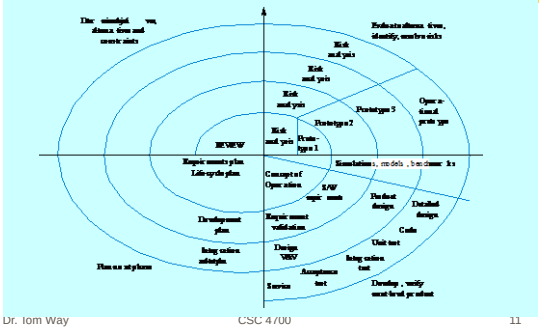
Incremental development



Process Iteration & Incremental Delivery

- System requirements ALWAYS evolve in the course of development
- Iteration can be applied to any of the generic process models
- Break down software into "releases", deliver incrementally (version 1.0, version 2.0, etc.)
- "Freeze" requirements during a release

Spiral model of the software process



Spiral development

- Process is represented as a spiral rather than as a sequence of activities with backtracking.
- Each loop in the spiral represents a phase in the process.
- No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required.
- Risks are explicitly assessed and resolved throughout the process.

The newer way

- Agile – it's Spiderman at the keyboard
- Extreme – it's like totally radical
- Scrum – what's rugby got to do with it?

The Agile Approach (1)

- Continuous delivery of software
- Cycle is weeks rather than months
- Working software is the principal measure of progress
- Even late changes in requirements are welcomed
- Close, daily, cooperation between business people and developers
- Face-to-face conversation is the best form of communication.

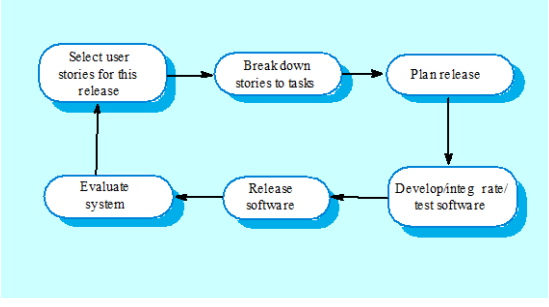
The Agile Approach (2)

- Projects are built around motivated individuals, who should be trusted
- Continuous attention to technical excellence and good design
- Simplicity
- Self-organizing teams
- Regular adaptation to changing circumstances
- From the *Agile Manifesto* (Google it)

Extreme programming

- Perhaps the best-known and most widely used agile method.
- Extreme Programming (XP) takes an 'extreme' approach to iterative development.
  - New versions may be built several times per day
  - New version delivered every 2 weeks
  - All tests run with each build, all must pass
  - Doesn't reinvent the wheel – use COTS whenever possible and affordable

The XP release cycle



Extreme programming practices 1

Incremental planning	Requirements are recorded on Story Cards and the Stories to be included in a release are determined by the time available and their relative priority. The developers break these Stories into development ŒTasksŒ
Small Releases	The minimal useful set of functionality that provides business value is developed first. Releases of the system are frequent and incrementally add functionality to the first release.
Simple Design	Enough de sign is carried out to meet the current requirements and no more.
Test first development	An automated unit test framework is used to write tests for a new piece of functionality before that functionality itself is implemented.
Refactoring	All developers are expected to refactor the code con tinuously as soon as possible code improvements are found. This keeps the code simple and maintainable.

Extreme programming practices 2

Pair Programming	Developers work in pairs, checking each other's work and providing the support to always do a good job.
Collective Ownership	The pairs of developers work on all areas of the system, so that no islands of expertise develop and all the developers own all the code. Anyone can change anything.
Continuous Integration	As soon as work on a task is complete it is integrated into the whole system. After any such integration, all the unit tests in the system must pass.
Sustainable pace	Large amounts of over-time are not considered acceptable as the net effect is often to reduce code quality and medium term productivity
On-site Customer	A representative of the end-user of the system (the Customer) should be available full time for the use of the XP team. In an extreme programming process, the customer is a member of the development team and is responsible for bringing system requirements to the team for implementation.

Problems with agile methods

- It can be difficult to keep the interest of customers who are involved in the process.
- Team members may be unsuited to the intense involvement that characterises agile methods.
- Prioritizing changes can be difficult where there are multiple stakeholders.
- Maintaining simplicity requires extra work.
- Contracts may be a problem as with other approaches to iterative development.

Testing in XP

- Test-first development
- For each and every component (class, module, whatever) you develop, add one or more tests at the same time
- Building means compiling the code and running all the tests, automatically
- Keeps software working all the time

Pair programming

- In XP, programmers work in pairs, sitting together to develop code.
- This helps develop common ownership of code and spreads knowledge across the team.
- It serves as an informal review process as each line of code is looked at by more than 1 person.
- It encourages refactoring as the whole team can benefit from this.
- Measurements suggest that development productivity with pair programming is similar to that of two people working independently.

SCRUM Approach

- Backlog – list of all of the tasks to get done
- Sprint – short iteration, get current backlog items done in this time
- Scrum – short, daily stand-up meeting
- Planning session – start of each sprint, plan which backlog items will be done
- Heartbeat retrospective – end of sprint, reflect about the past sprint

SCRUM stuff

- Scrum Master - removes impediments to the ability of the team to deliver the sprint goal, not the team leader
- Self organizing teams – magically everybody gets organized
- Easily adapt to change – major benefit