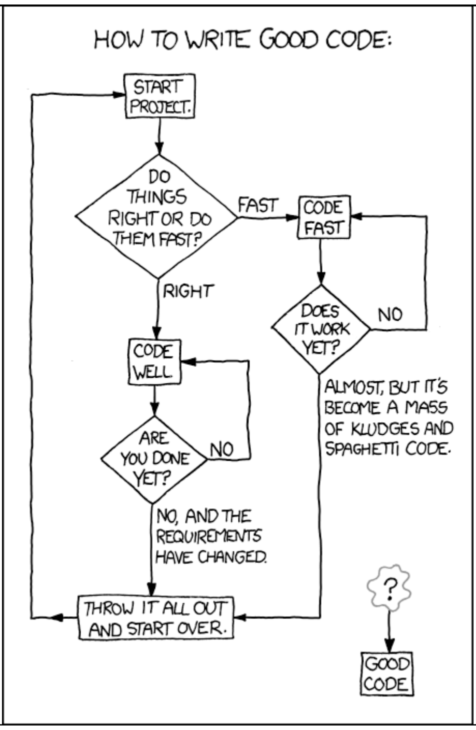
Lecture 7: Software Processes

- ➜ What is a Software Development Process?
- ➜ The Lifecycle of a Software Project
- ➜ Agile vs. Disciplined
- ➜ Some common approaches:
  - ➷ RUP, SCRUM, XP, ICONIX,…
- ➜ Where UML fits in (next lecture)


HOW TO WRITE GOOD CODE:
START PROJECT.
DO THINGS RIGHT OR DO THEM FAST?
FAST
CODE FAST
DOES IT WORK YET?
NO
ALMOST, BUT IT'S BECOME A MASS OF KLUDGES AND SPAGHETTI CODE.
RIGHT
CODE WELL
ARE YOU DONE YET?
NO
NO, AND THE REQUIREMENTS HAVE CHANGED.
THROW IT ALL OUT AND START OVER.
?
GOOD CODE

Source: xkcd 844

Project Types

Reasons for initiating a software development project

- Problem-driven: competition, crisis,…
- Change-driven: new needs, growth, change in business or environment,…
- Opportunity-driven: exploit a new technology,…
- Legacy-driven: part of a previous plan, unfinished work, …

Relationship with Customer(s):

- Customer-specific - one customer with specific problem
  - May be another company, with contractual arrangement
  - May be a division within the same company
- Market-based - system to be sold to a general market
  - In some cases the product must generate customers
  - Marketing team may act as substitute customer
- Community-based - intended as a general benefit to some community
  - E.g. open source tools, tools for scientific research
  - Usually: funder ≠ customer (if funder has no stake in the outcome)
- Hybrid (a mix of the above)

Project Context

What is the current (old) system?

- There is *always* an existing system!
  - May just be a set of ad hoc workarounds for the problem
- Studying it is important:
  - If we want to avoid the weaknesses of the old system…
  - …while preserving what the stakeholders like about it

Use pre-existing components?

- Benefits:
  - Can dramatically reduce development cost
  - Easier to decompose the problem if some sub-problems are already solved
- Tension:
  - Solving the real problem vs. solving a known problem (with ready solution)

Will it be part of a product family?

- Vertical families: e.g. 'basic', 'deluxe' and 'pro' versions of a system
- Horizontal families: similar systems used in related domains
  - Typically based on a common architecture (or just shared software assets)

Lifecycle of an Engineering Project

Lifecycle models

- Useful for comparing projects in general terms
- Not enough detail for project planning
- Examples:
  - Sequential models: Waterfall, V model
  - Phased Models: Incremental, Evolutionary
  - Iterative Models: Spiral

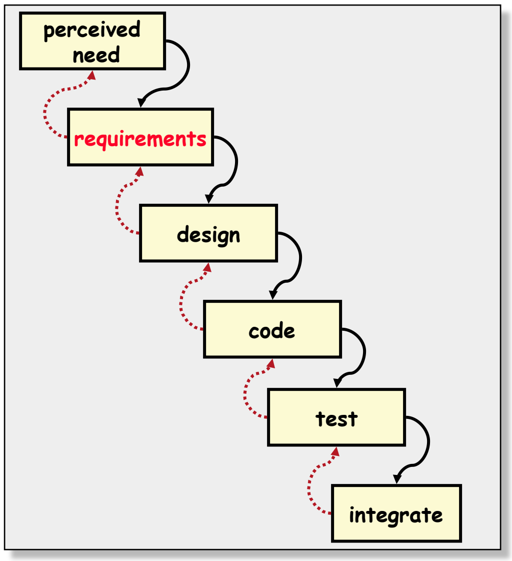
Process Models

- Used for capturing and improving the development process
- Detailed guidance on steps and products of each step

Process Frameworks

- Patterns and principles for designing a specific process for your project

Waterfall Model



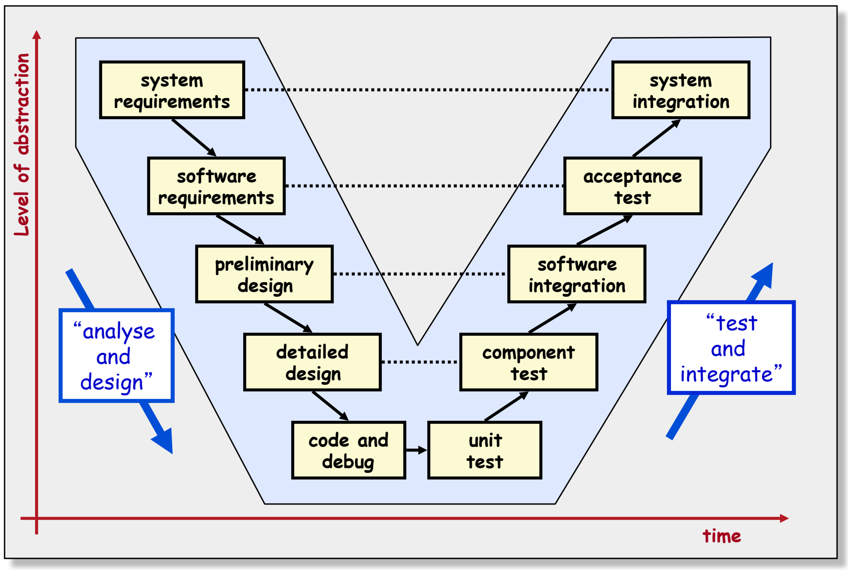
View of development:

- a process of stepwise refinement
- largely a high level management view

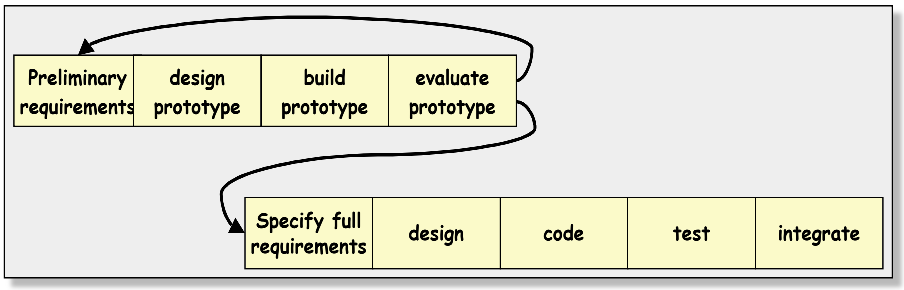
Problems:

- Static view of requirements - ignores volatility
- Lack of user involvement once specification is written
- Unrealistic separation of specification from design
- Doesn't accommodate prototyping, reuse, etc.

V-Model



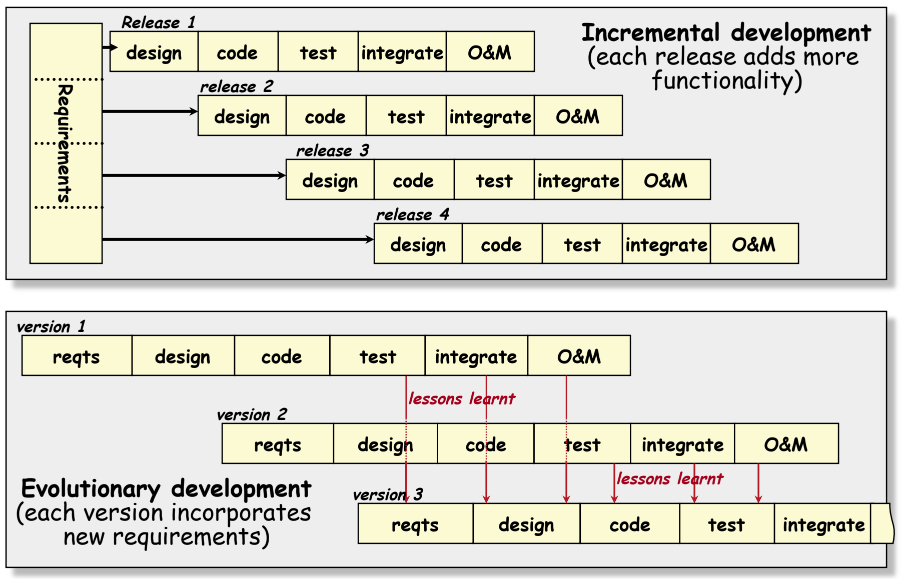
Prototyping lifecycle



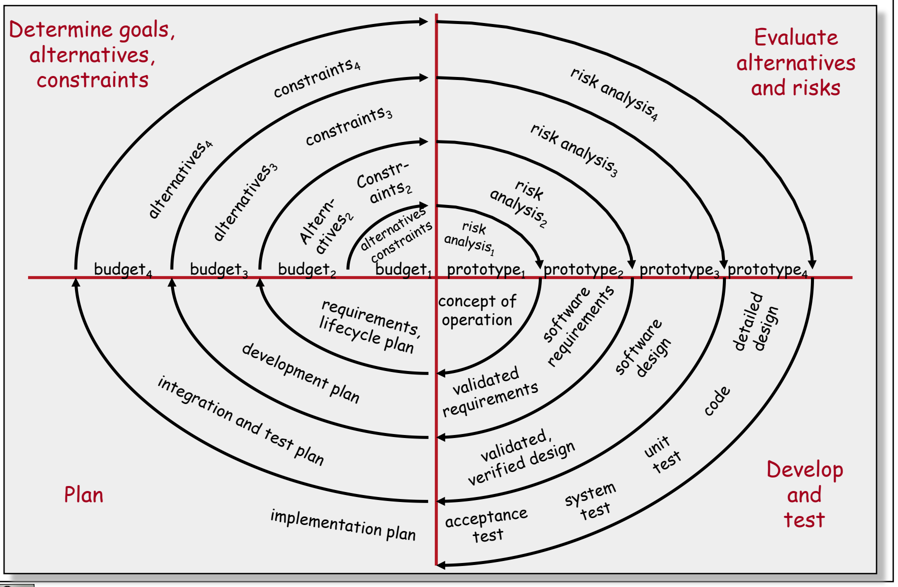
Prototyping is used for:
- understanding the requirements for the user interface
- examining feasibility of a proposed design approach
- exploring system performance issues

Problems:
- users treat the prototype as the solution
- a prototype is only a partial specification

Phased Lifecycle Models



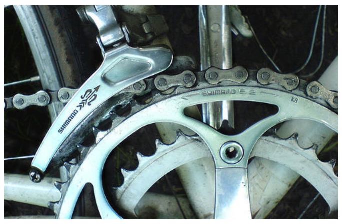
The Spiral Model



Why the emphasis on “process”?

Quality in Use
(What’s the end-user’s experience?)

External Quality Attributes
(Does it pass all the tests?)



Internal Quality Attributes
(Is it well-designed?)

Process Quality
(Is it assembled correctly?)

“Agile” vs “Sturdy”

Iterative	⟷	**Planned**
Small increments	⟷	**Analysis before design**
Adaptive planning	⟷	**Prescriptive planning**
Embrace change	⟷	**Control change**
Innovation and exploration	⟷	**High ceremony**
Trendy	⟷	**Traditional**
Highly fluid	⟷	**Upfront design / architecture**
Feedback driven	⟷	**Negotiated requirements**
Individuals and Interactions	⟷	**Processes and Tools**
Human communication	⟷	**Documentation**
Small teams	⟷	**Large teams**

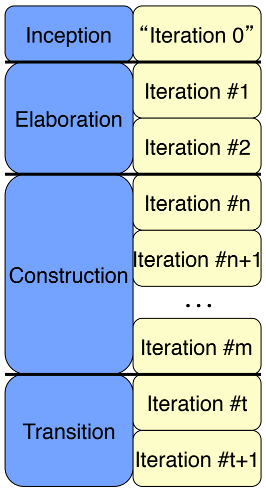
Rational Unified Process (RUP)

Inception
- **Establish Scope**
- **Build a business case**
- **Get stakeholder buy-in**

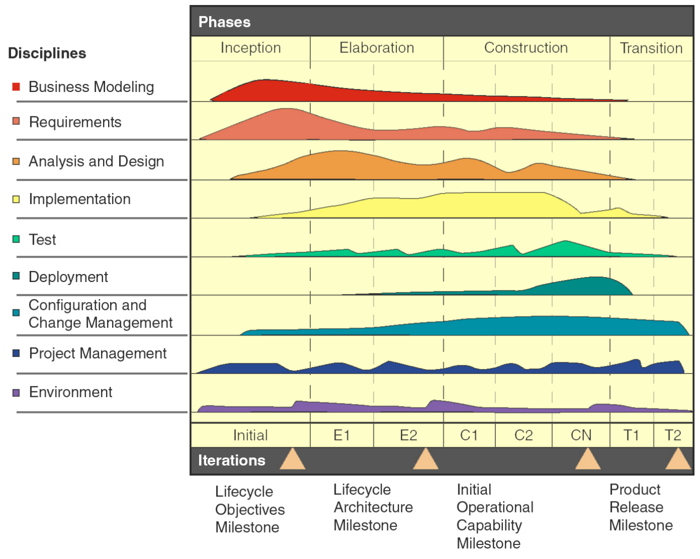
Elaboration
- **Identify and manage risks**
- **Build an executable architecture**
- **Focus only on high risk items**

Construction
- **Iteratively build operational version**
- **Develop support docs and training materials**

Transition
- **Fine-tune**
- **Resolve configuration, installation and usability issues**



RUP Activities



SCRUM

Sprint - 30 day iteration

- Starts with 1/2 day planning meeting
- Starts with Prioritized Product Backlog (from product owner)
- Builds a Sprint Backlog - items to be done in this sprint
- 29 days of development
- 1/2 day Sprint review meeting - inspect product, capture lessons learnt

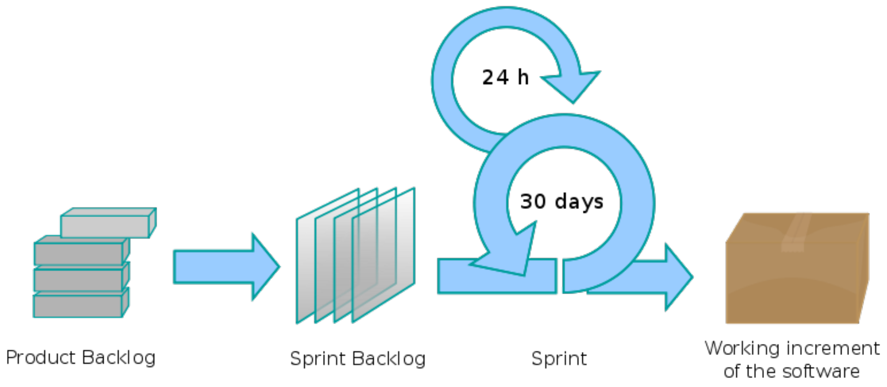
Daily Scrum

- 15 minute team meeting each day.
- Each team member answers:
  - What have you done since last meeting?
  - What will you do between now and the next meeting?
  - What obstacles stood in the way of doing work?
- Scrum master keeps meeting on track

Scrum teams

- Cross-functional, 7 (±2) members
- Teams are self-organising

Scrum Process



Source: wikipedia

Extreme Programming

Fine Scale Feedback

- Pair Programming
- Planning Game
- Test-driven Development
- Whole team (customer part of team)

Continuous Process

- Continuous Integration
- Design Improvement (refactoring)
- Small Releases

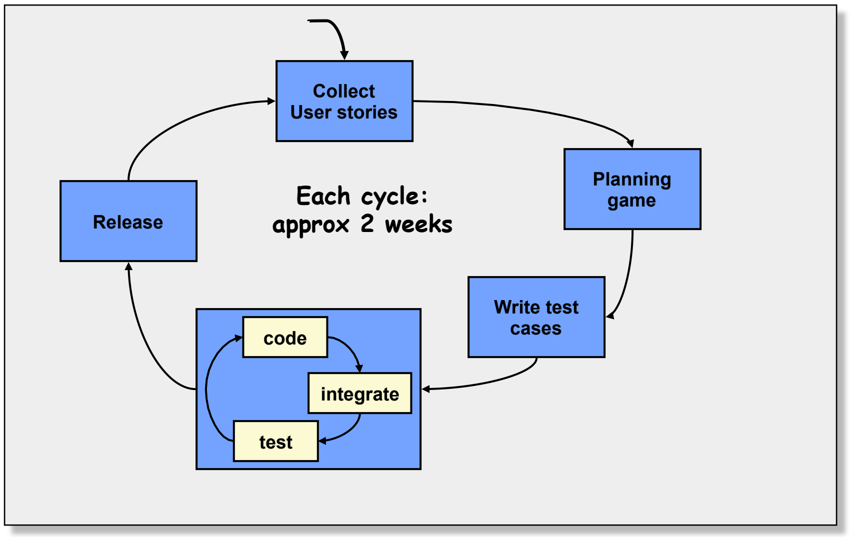
Shared Understanding

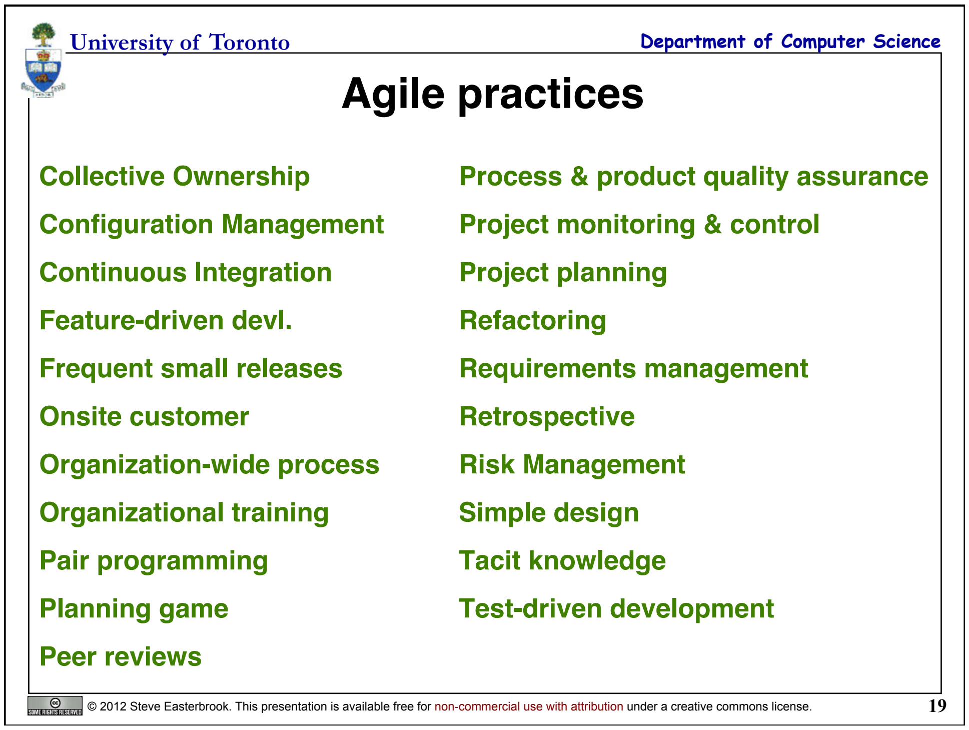
- Coding Standards
- Collective Code Ownership
- Simple Design
- System Metaphor

Programmer Welfare

- Sustainable pace (40 hour week)

Extreme Programming





Agile practices

Collective Ownership

Configuration Management

Continuous Integration

Feature-driven devl.

Frequent small releases

Onsite customer

Organization-wide process

Organizational training

Pair programming

Planning game

Peer reviews

Process & product quality assurance

Project monitoring & control

Project planning

Refactoring

Requirements management

Retrospective

Risk Management

Simple design

Tacit knowledge

Test-driven development