Imperial College
London

Software Engineering Practice

Agile Methods and Prototypes

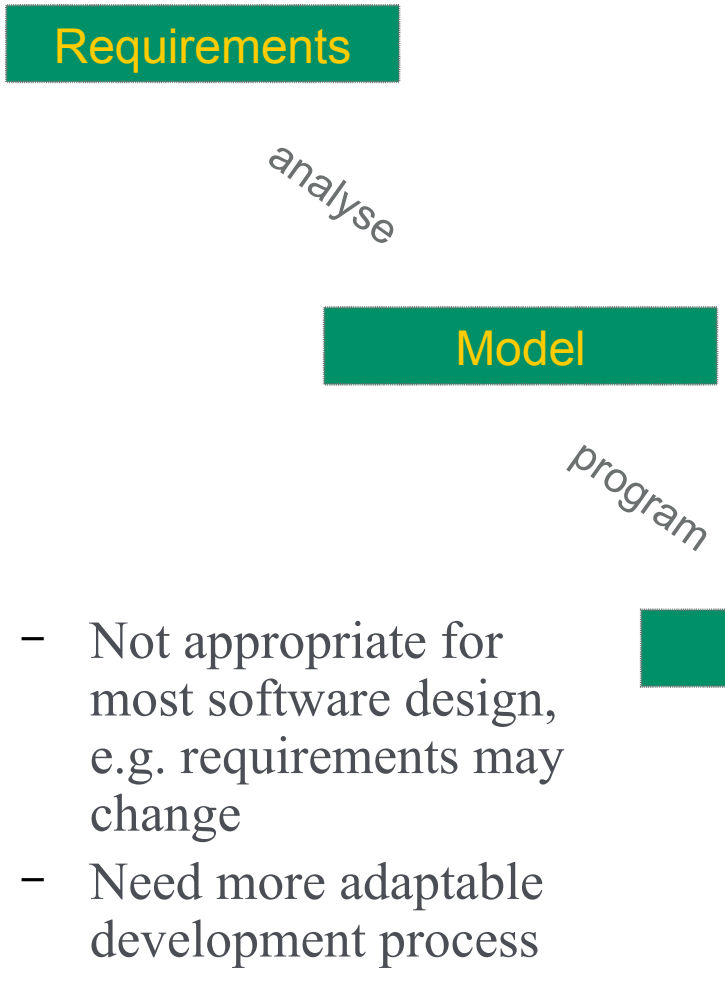
Lecture Overview

- Software Development Processes
  - The Waterfall Model
  - Iterative Development/Spiral Model
- Agile Development Methodologies
  - Scrum
  - XP
- Prototyping

Software Engineering

- High degree of novelty and change, non-linear progress
  - Hard to estimate costs, necessary resources, completion times
  - Difficult to capture requirements

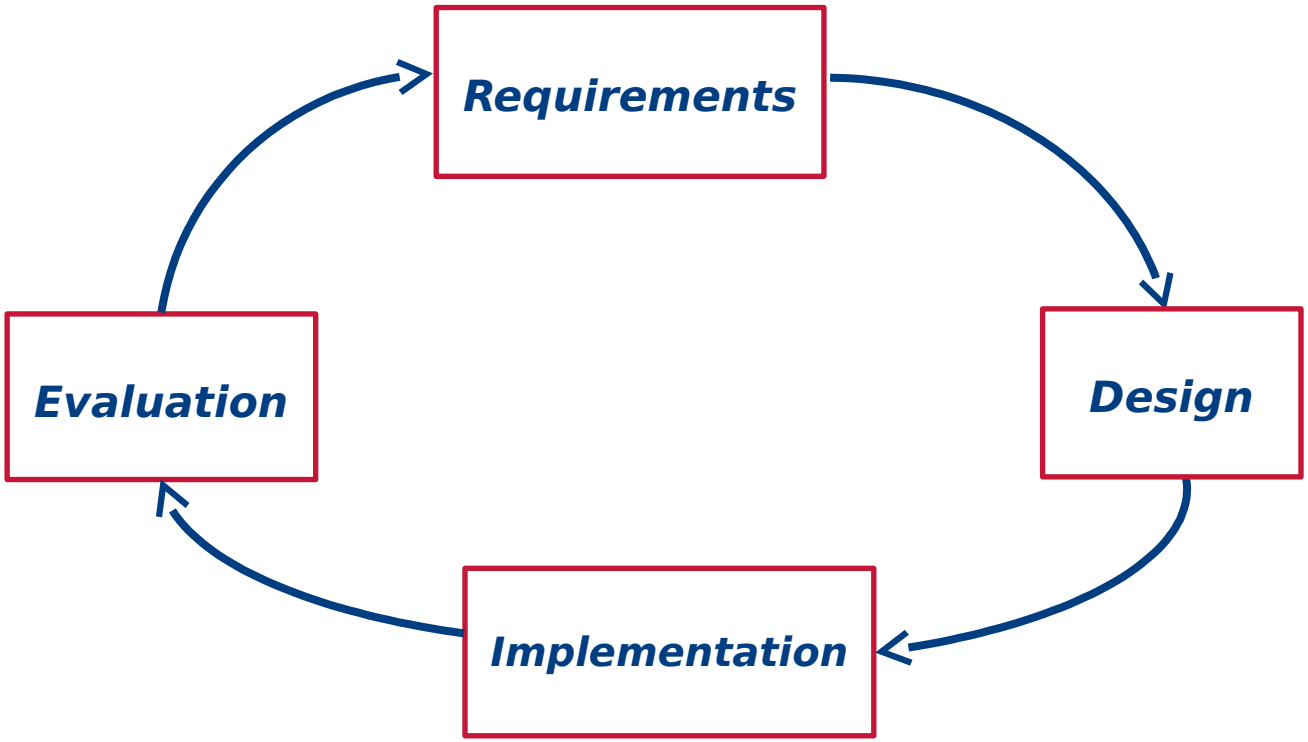
The Waterfall Model



1) Capture requirements
2) Construct model / design
3) Implement code
4) Deploy system… *and probably fail*

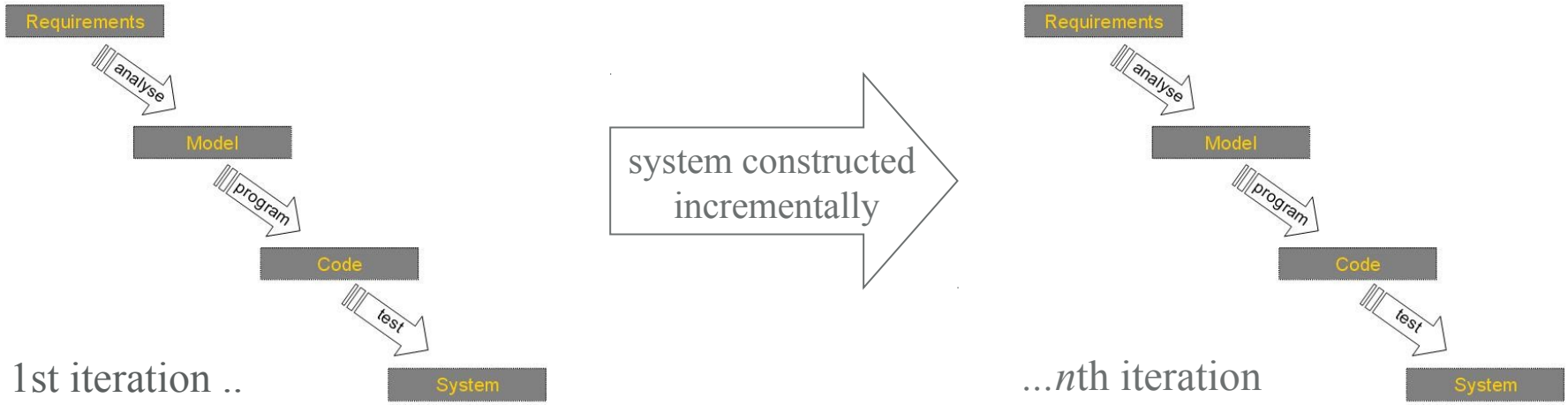
- Not appropriate for most software design, e.g. requirements may change
- Need more adaptable development process

Iterative Development



Imperial College London

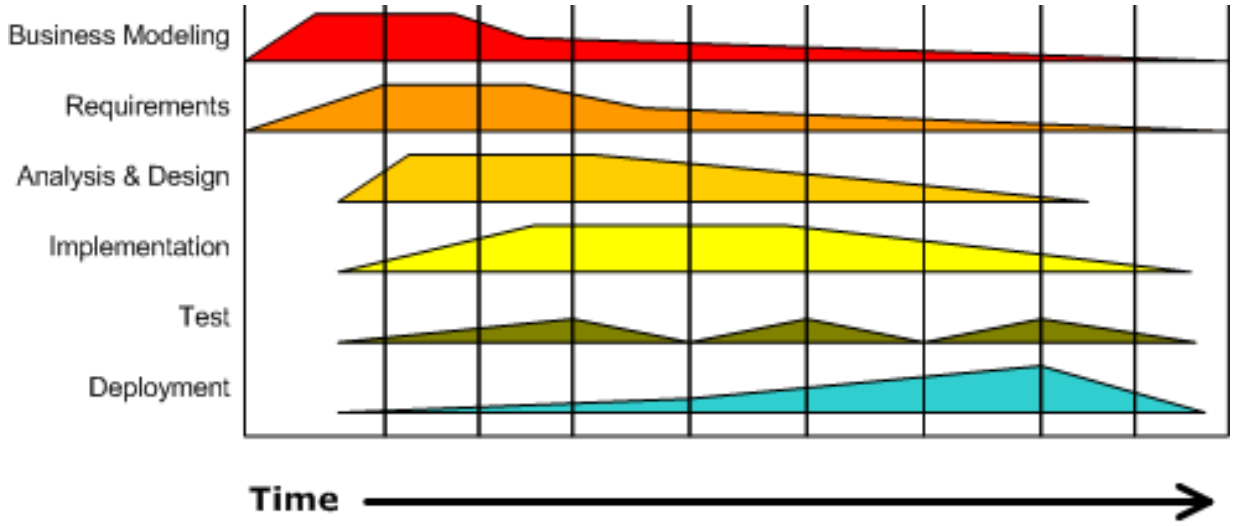
Iterative Development



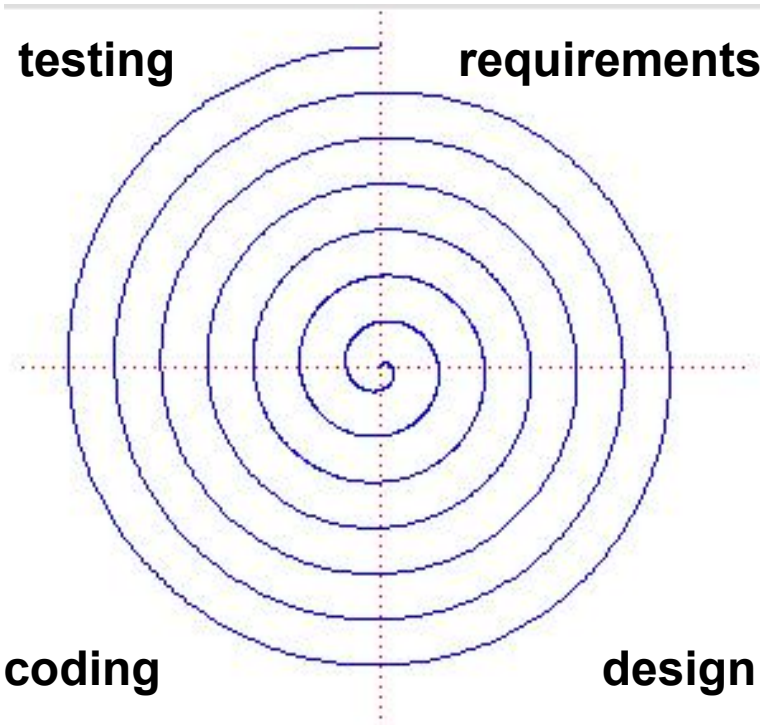
- Life-cycle consists of several iterations in sequence
- Each iteration is a self-contained mini-project
  - Requirements, design, programming, testing, etc
- Goal of each iteration is an executable system
  - Integration and testing of all software
- Multiple waterfall cycles??

Iterative Development

- Typically more focus on requirements and design in earlier iterations
- More implementation, testing in later iterations



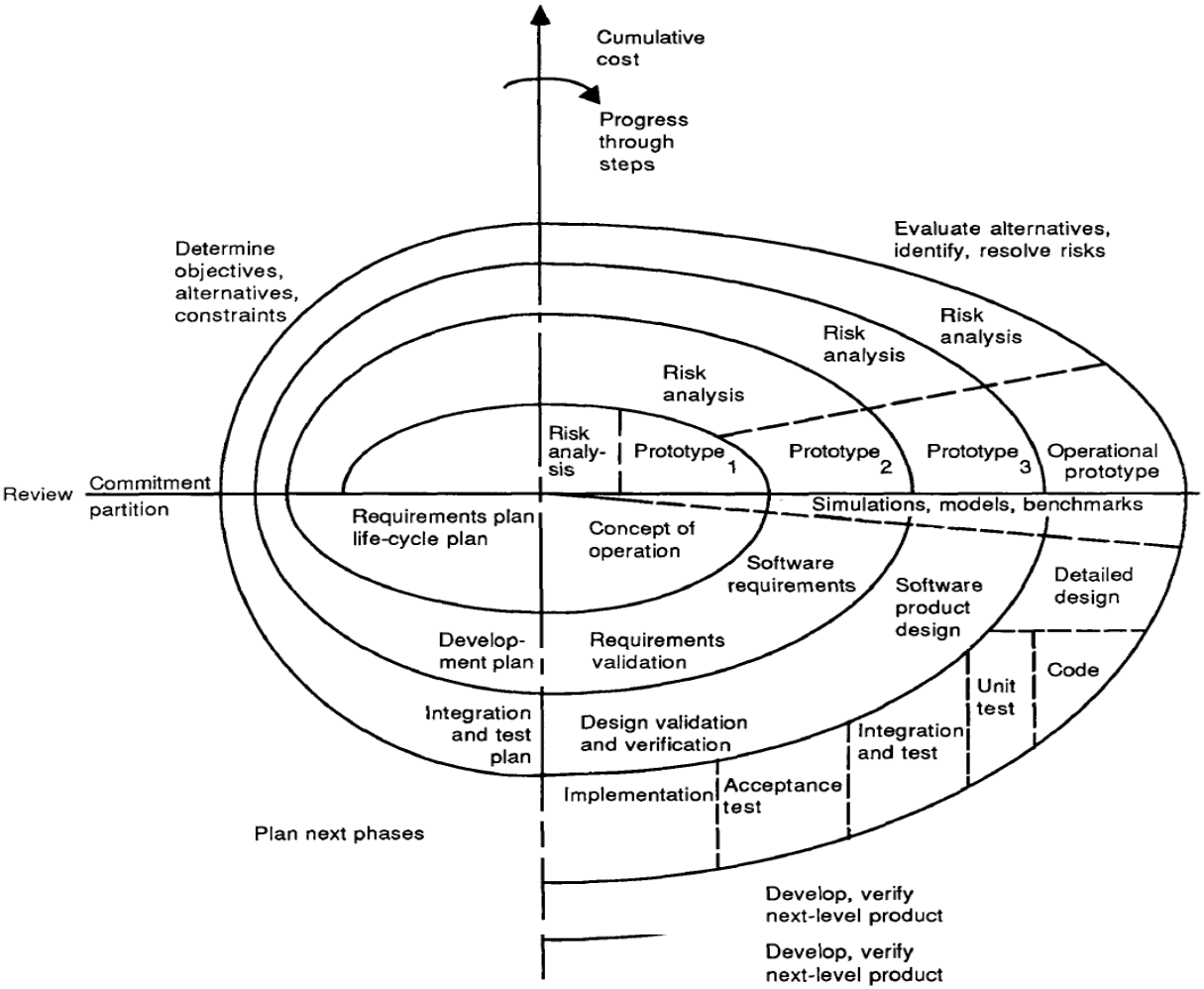
Spiral Model



- Early iterations are shorter, cheaper, riskier
- Duration of an iteration depends on the project
- For your projects, iterations ranging from a few days to a few weeks would be suitable

Imperial College London

Spiral Model



Reading: ***A Spiral Model of Software Development and Enhancement*** by **Barry W. Boehm**

Agile Development

- An ideology for software development emphasising
  - Working software vs. models and documentation
  - Customer involvement and feedback vs. fixed initial contract
  - Response to change vs. long-term planning
  - Refactoring vs. lengthy design
- More flexible, dynamic, *agile*

- Some drawbacks
  - Lack of stable design not always a good thing (efficiency?)
  - Does not scale well to large teams

Agile Methods

- Agile methods in practice include
  - *Scrum*
  - *eXtreme Programming (XP)*
  - *Rational Unified Process (RUP)*
  - *Dynamic Systems Development Method*

Agile Methods - Common Sense

- Methods are introduced
  - Partly to give you a brief overview of agile software development practices
  - And partly to give you ideas about how you should conduct your group projects
- Use common sense to evaluate what does and doesn't work for your group and your project
- Methods not presented as things to be followed to the letter, but in spirit

Scrum

- Inspired by rugby scrum
  - Method of restarting play in rugby
  - A rapidly changing, small-group effort to achieve a goal
  - Might be well suited to your group projects

Scrum: Key Practices

- Self-directed, self-organising teams
- Product Backlog
- Short (e.g. 1-4 wk) iterations, called sprints
  - Basic unit of development
  - Initial sprint planning (what work has to be done)
  - Daily meetings with special questions
  - Sprint review and reflection

Product Backlog

- Ordered list of required features, non-functional requirements, bug fixes etc
  - What needs to be done to deliver the desired product
- Broken down into tasks with assigned:
  - Priority levels (list order)
  - Time needed for completion
- Editable by anyone involved
  - Clients (priorities)
  - Developers (estimated effort)

Sprints

- Sprint Planning Meeting
  - At the beginning of each sprint
  - Prepares the current sprint backlog
- Sprint Backlog
  - List of tasks that team commits to completing in current sprint
  - Tasks are kept short (e.g. 4-6 hours)
  - Team members pick tasks from the backlog
  - Backlog frozen during sprint

Daily Scrum Meeting

- Heartbeat of the project
  - Every workday at same time and place
- Everybody can come (e.g. clients, management)
  - But only team members can talk
- Each member answers these questions
  - What have you done since the last Scrum meeting?
  - What will you do between now and the next meeting?
  - What is getting in the way of the goals?
- Discussion is brief and focuses only on these questions
- Update and display sprint and product backlogs at each meeting
  - Use these to track estimated completion time

Sprints (cont)

- Sprint Review Meeting
  - At the end of each sprint
  - Review the work done in sprint
  - Demo product to stakeholders
- Sprint Retrospective Meeting
  - Team reflects on the past sprint
  - What could be improved in the next sprint?

XP Programming

- Key values:
  - Communication
  - Feedback
  - Simplicity

XP: Communication

- Communication is essential
  - Among developers
  - Between developers and client
  - Communication favoured over documentation
- Pair programming
  - Two developers working together at a single computer
    - As equals
  - Code reviewing becomes part of the development process
- Collective code ownership
  - Avoids finger pointing
  - Everyone can fix code and introduce errors (testing!)

XP: Feedback

- Client involvement and feedback is essential

- Acceptance cycle:
  - Client writes user stories
  - User stories determine acceptance tests
  - Tests determine whether code does what client wants

- Test-driven development
  - Tests written before product code
  - Tests play the role of formal model/specification
  - Work is done when no tests fail

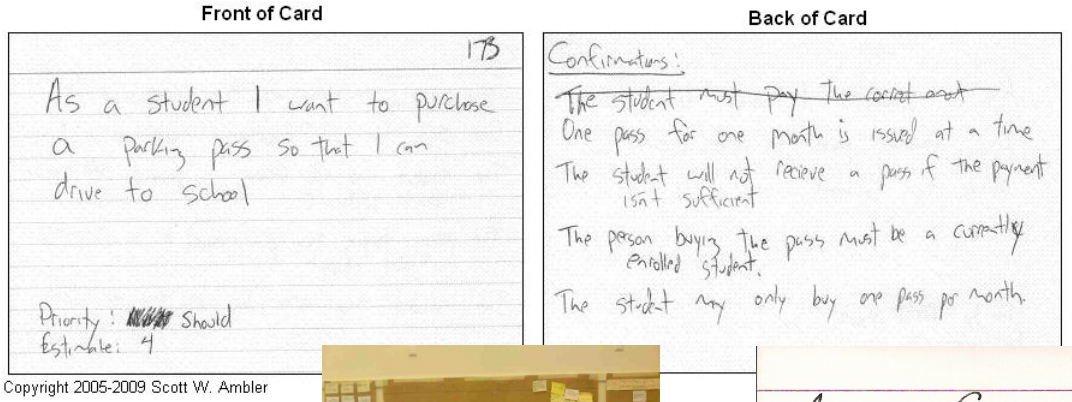
User Stories

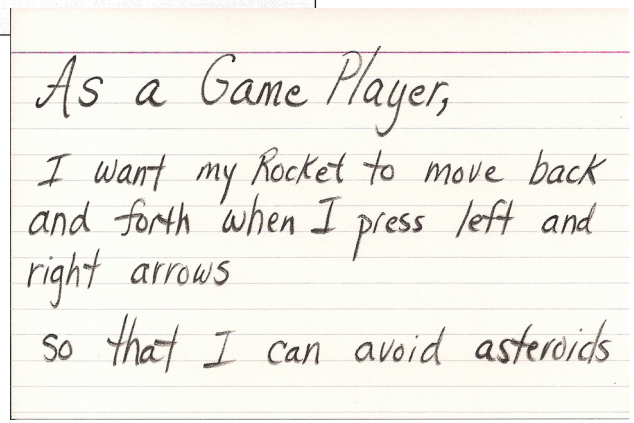
- User stories are short sentences capturing the desired functionality of the system in terms of what users (need to) do
  - As a <role>, I want <goal/desire>
  - As a <role>, I want <goal/desire> so that <benefit>
  - In order to <receive benefit> as a <role>, I want <goal/desire>
  - As <who> <when> <where>, I <what> because <why>

Imperial College London

User Stories

- User stories are short sentences capturing the desired functionality of the system in terms of what users (need to) do





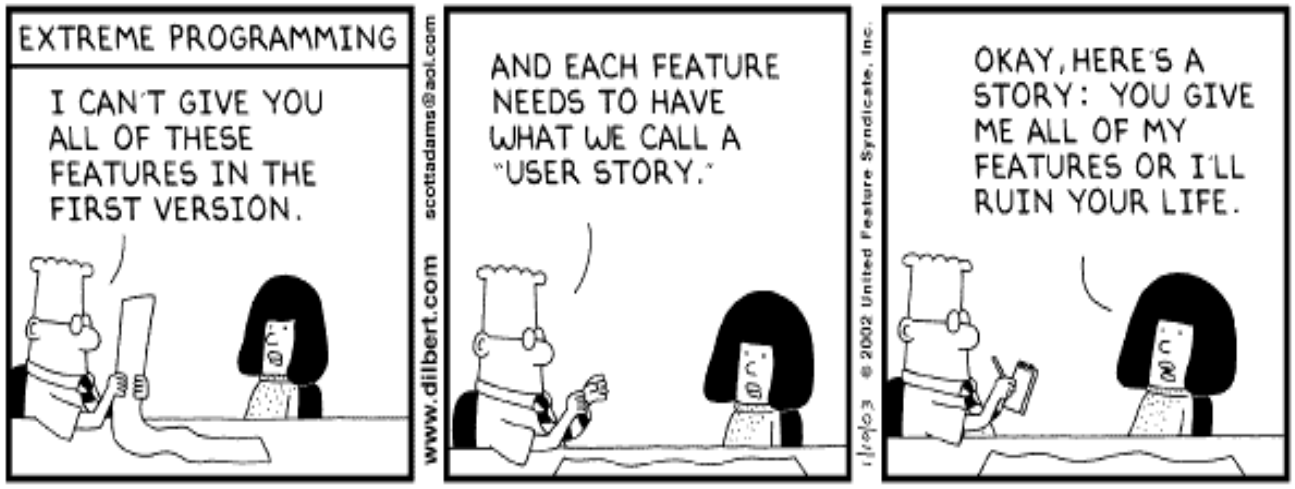
XP: Simplicity

- Things to avoid:
  - Speculative design about future changes
  - Generalised components that are not currently needed
- Refactoring used frequently
  - Compensates for minimal initial design
- Common sense should dictate trade-off between design and refactoring

XP: Process

- Planning
  - Client creates user stories
  - User stories are broken into tasks
  - Time needed to complete each task is estimated
  - Developers pick tasks
- Small, frequent releases
  - Frequent communication among developers
  - Continuous customer involvement
  - Test-driven development
  - Frequent refactoring

XP: What Can Go Wrong



XP: What Can Go Wrong

- Tests
  - Not written first or not integrated in process
- Insufficient refactoring
  - Low-key design effort needs to be compensated by aggressive refactoring
  - Typically 25% of coding effort
- Pair programming
  - Pair members don't work as equals
  - Pair consists of two inexperienced programmers

Agile Development: Summary

- Agile methods favor:
  - Iterative development
  - Communication favoured over documentation
  - Refactoring favoured over initial design
  - Frequent feedback from client
  - Working software
- Use common sense to evaluate what does and doesn't work for your group and your project