# Lecture 3: What is Engineering?

- What is engineering about?
  - Engineering vs. Science
  - Devices vs. Systems
  - How is software engineering different?
  - Engineering as a profession
- Engineering Projects
  - Project Management
  - Project Initiation
- Project Lifecycles
  - Software Engineering lifecycles: Waterfalls, spirals, etc
  - Requirements Lifecycles





# Engineering vs. Science

- Traditional View:

| Scientists... | Engineers... |
|---|---|
| create knowledge | apply that knowledge |
| study the world as it is | seek to change the world |
| are trained in scientific method | are trained in engineering design |
| use explicit knowledge | use tacit knowledge |
| are thinkers | are doers |

- More realistic View

| Scientists... | Engineers... |
|---|---|
| create knowledge | create knowledge |
| are problem-driven | are problem-driven |
| seek to understand and explain | seek to understand and explain |
| design experiments to test theories | design devices to test theories |
| prefer abstract knowledge | prefer contingent knowledge |
| but rely on tacit knowledge | but rely on tacit knowledge |

**Both involve a mix of design and discovery**

# What is engineering?

"Engineering is the development of cost-effective solutions to practical problems, through the application of scientific knowledge"

## "…Cost-effective…"

- Consideration of design trade-offs, esp. resource usage
- Minimize negative impacts (e.g. environmental and social cost)

## "… Solutions …"

- Emphasis on building devices

## "… Practical problems …"

- solving problems that matter to people
- improving human life in general through technological advance

## "… Application of scientific knowledge …"

- Systematic application of analytical techniques






# Devices vs. Systems

## Normal design:

- Old problems, whose solutions are well known
  - Engineering codifies standard solutions
  - Engineer selects appropriate methods and technologies
- Design focuses on well understood devices
  - Devices can be studied independent of context
  - Differences between the mathematical model and the reality are minimal

## Radical design:

- Never been done, or past solutions have failed
  - Often involves a very complex problem
- Bring together complex assemblies of devices into new systems
  - Such systems are not amenable to reductionist theories
  - Such systems are often soft: no objective criteria for describing the system

## Examples:

- Most of Computer Engineering involves normal design
- All of Systems Engineering involves radical design (by definition!)
- Much of Software Engineering involves radical design (soft systems!)

# Is software different?

- **Software is different!**
  - software is invisible, intangible, abstract
    - its purpose is to configure some hardware to do something useful
  - there are no physical laws underlying software behaviour
  - there are no physical constraints on software complexity
  - software never wears out
    - …traditional reliability measures don't apply
  - software can be replicated perfectly
    - …no manufacturing variability
- **Software Myths:**
  - Myth: Cost of software is lower than cost of physical devices
  - Myth: Software is easy to change
  - Myth: Computers are more reliable than physical devices
  - Myth: Software can be formally proved to be correct
  - Myth: Software reuse increases safety and reliability
  - Myth? Computers reduce risk over mechanical systems





# Professional Responsibility

- **ACM/IEEE code of ethics:**
  - PUBLIC - act consistently with the public interest.
  - CLIENT AND EMPLOYER - act in a manner that is in the best interests of your client and employer, consistent with the public interest.
  - PRODUCT - ensure that your products and related modifications meet the highest professional standards possible.
  - JUDGEMENT - maintain integrity and independence in your professional judgment.
  - MANAGEMENT - subscribe to and promote an ethical approach to the management of software development and maintenance.
  - PROFESSION - advance the integrity and reputation of the profession consistent with the public interest.
  - COLLEAGUES - be fair to and supportive of your colleagues.
  - SELF - participate in lifelong learning and promote an ethical approach to the practice of the profession.
- **Of particular relevance in RE:**
  - Competence - never misrepresent your level of competence
  - Confidentiality - respect confidentiality of all stakeholders
  - Intellectual property rights - respect protections on ideas and designs
  - Data Protection - be aware of relevant laws on handling personal data

# Project Management

- **A manager can control 4 things:**
  - **Resources** (can get more dollars, facilities, personnel)
  - **Time** (can increase schedule, delay milestones, etc.)
  - **Product** (can reduce functionality - e.g. scrub requirements)
  - **Risk** (can decide which risks are acceptable)
- **To do this, a manager needs to keep track of:**
  - **Effort** - How much effort will be needed? How much has been expended?
  - **Time** - What is the expected schedule? How far are we deviating from it?
  - **Size** - How big is the planned system? How much have we built?
  - **Defects** - How many errors are we making? How many are we detecting?
    - And how do these errors impact quality?
- **Initially, a manager needs good estimates**
  - …and these can only come from a thorough analysis of the problem.

You cannot control that which you cannot measure!





# 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
    - funder [1] customer (if funder has no stake in the outcome)
  - Hybrid (a mix of the above)

# Project Context

## ➲ Existing System

- There is nearly 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

## ➲ Pre-Existing Components

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

## ➲ Product Families

- Vertical families: e.g. 'basic', 'deluxe' and 'pro' versions of a system
- Horizontal families: similar systems used in related domains
  - Need to define a common architecture that supports anticipated variability





# Lifecycle of an Engineering Project

## ➲ Lifecycle models

- Useful for comparing projects in general terms
- Not enough detail for project planning

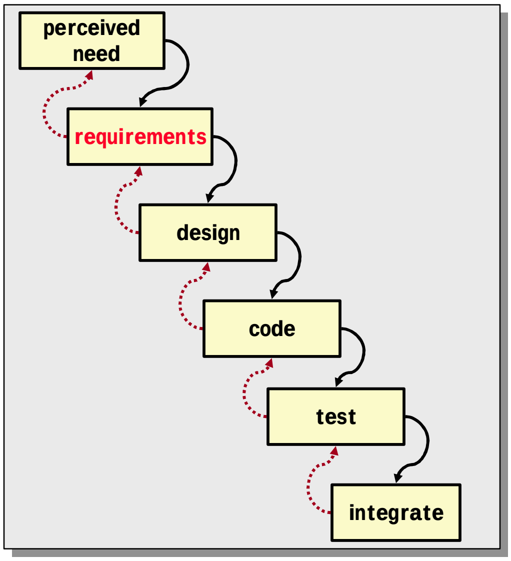
## ➲ Examples:

- Sequential models: Waterfall, V model
- Rapid Prototyping
- Phased Models: Incremental, Evolutionary
- Iterative Models: Spiral
- Agile Models: eXtreme Programming

## ➲ Comparison: Process Models

- Used for capturing and improving the development process

# Waterfall Model

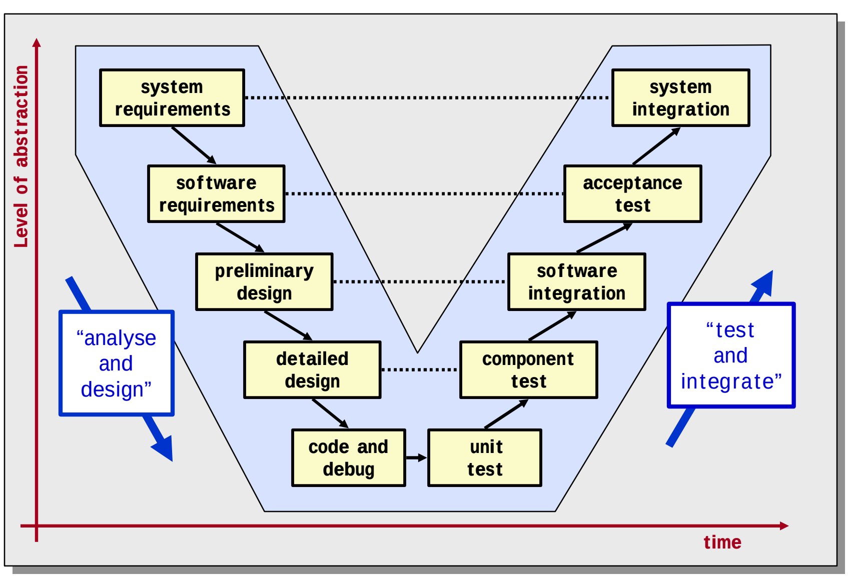



- View of development:
  - a process of stepwise refinement
  - largely a high level management view
- Problems:
  - Static view of requirements - ignores volatility
  - Lack of user involvement once specification is written
  - Unrealistic separation of specification from design
  - Doesn't accommodate prototyping, reuse, etc.

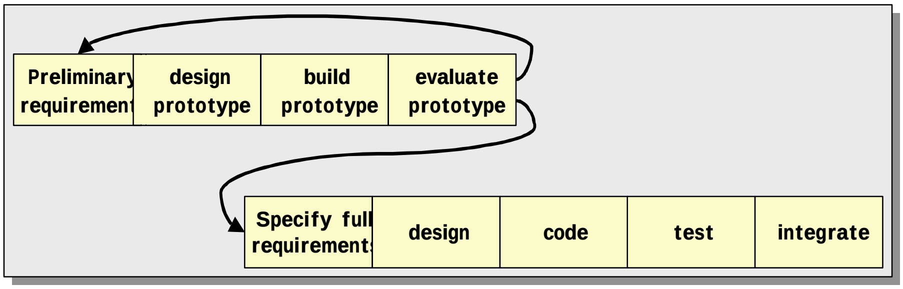




# V-Model



# Prototyping lifecycle

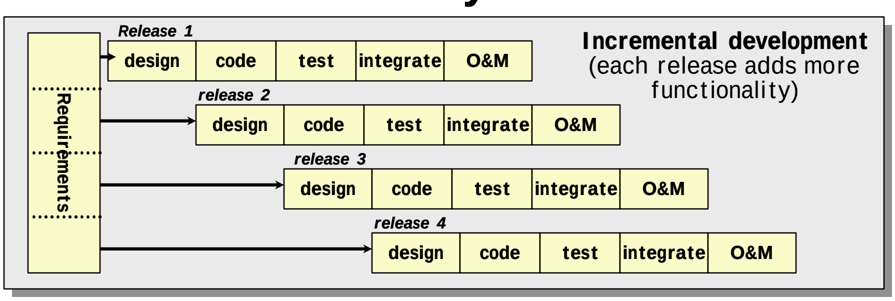


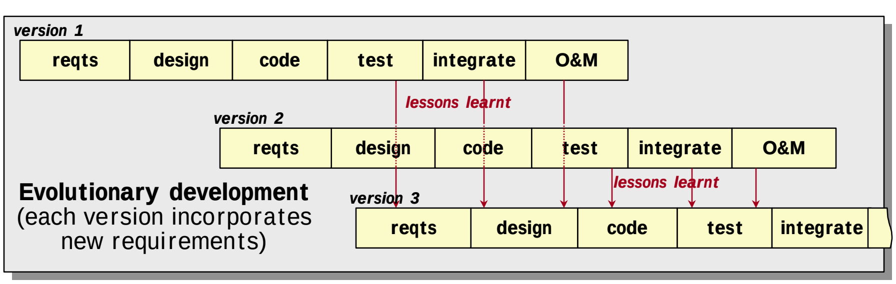

- **Prototyping is used for:**
  - understanding the requirements for the user interface
  - examining feasibility of a proposed design approach
  - exploring system performance issues
- **Problems:**
  - users treat the prototype as the solution
  - a prototype is only a partial specification






# Phased Lifecycle Models

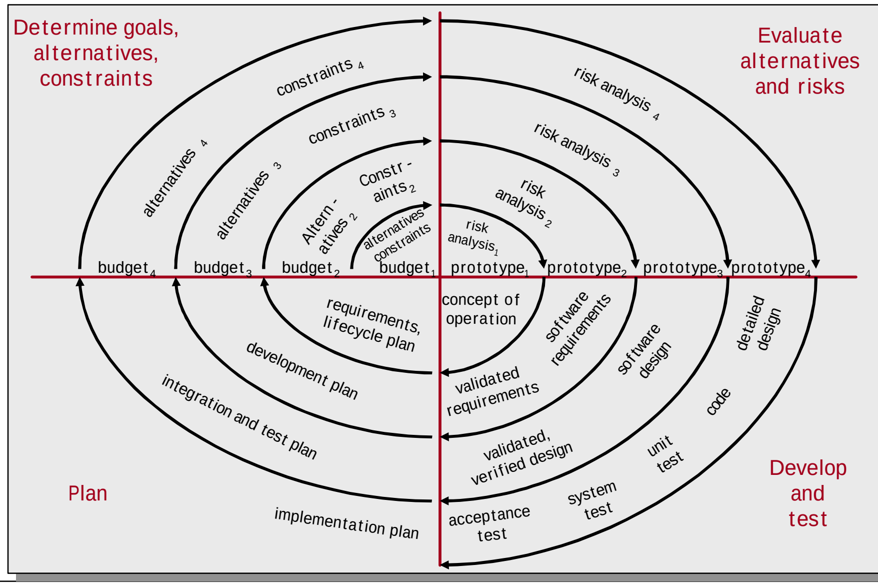




*Source:* Adapted from Dorfman, 1997, p10

# The Spiral Model





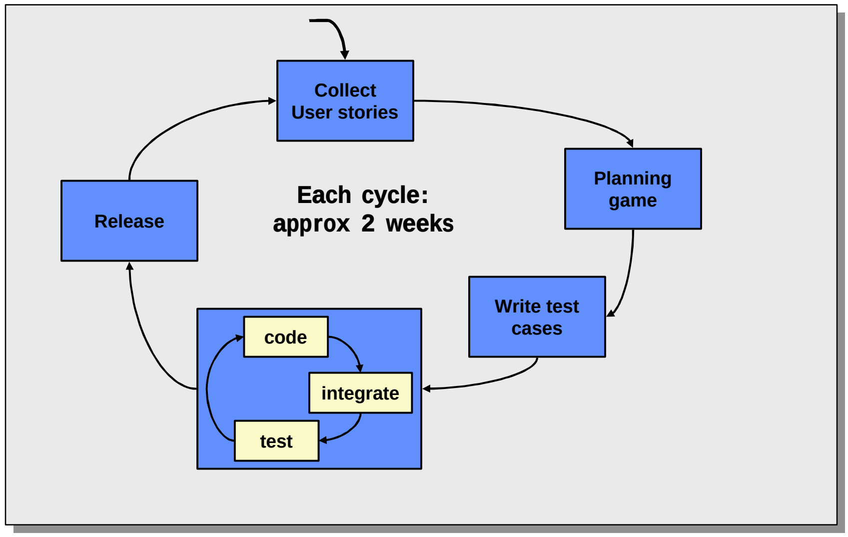



# Agile Models

- **Basic Philosophy**
  - Reduce communication barriers
    - Programmer interacts with customer
  - Reduce document-heavy approach
    - Documentation is expensive and of limited use
  - Have faith in the people
    - Don't need fancy process models to tell them what to do!
  - Respond to the customer
    - Rather than focusing on the contract
- **Weaknesses**
  - Relies on programmer's memory
    - Code can be hard to maintain
  - Relies on oral communication
    - Mis-interpretation possible
  - Assumes single customer representative
    - Multiple viewpoints not possible
  - Only short term planning
    - No longer term vision

**E.g. Extreme Programming**
- Instead of a requirements spec, use:
  - User story cards
  - On-site customer representative
- Pair Programming
- Small releases
  - E.g. every three weeks
- Planning game
  - Select and estimate user story cards at the beginning of each release
- Write test cases before code
- The program code is the design doc
  - Can also use CRC cards (Class-Responsibility-Collaboration)
- Continuous Integration
  - Integrate and test several times a day

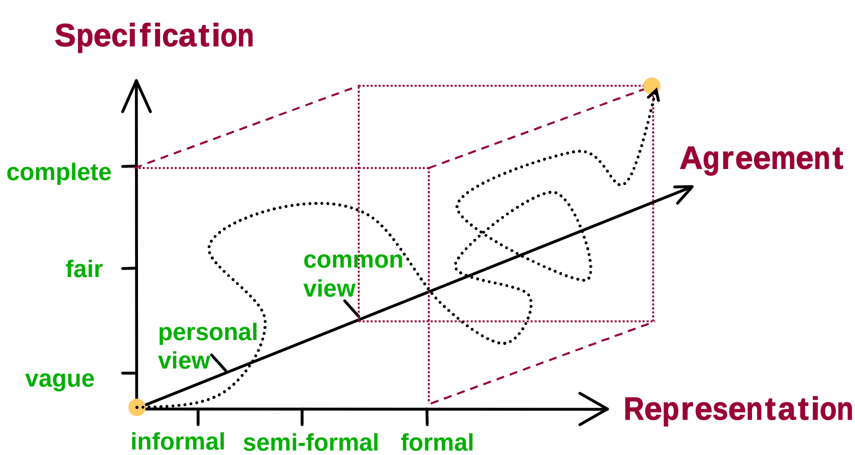
# Extreme Programming



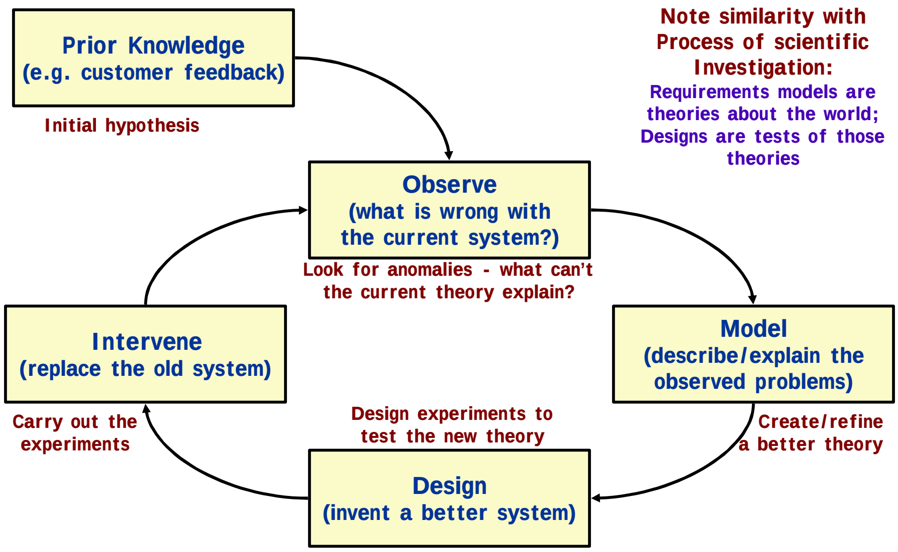





# Is there a "Requirements Lifecycle"



# Inquiry Cycle








# Summary

- **What is engineering?**
  - Not that different from science
  - Greater awareness of professional responsibility
    - because of immediate scope for harm to the public
  - Systems and Software Engineering involve radical design
- **Engineering Projects**
  - You cannot control that which you cannot measure
    - ...and many important measures are derived from initial problem analysis
  - Constraints:
    - Is there a customer?
    - Existing system / existing components / existing product family
- **Project Lifecycles**
  - Useful for comparing projects in general terms
  - Represent different philosophies in software development
  - Requirements evolve through their own lifecycles too!