SWE 530:
Software Design Process
Design in the Software Development Process

Dr. H. Birkan YILMAZ
Department of Computer Engineering
Boğaziçi University

(birkan.yilmaz@boun.edu.tr)

Adapted from slides of Dr. Albert Ali Salah

Recap

- The complexity of the model-building processes for software systems, with their need to consider **static** forms as well as the **dynamic** behaviour of the eventual system;
- The influence of the **invisible nature of software** upon any attempts to describe it;
- The need for **domain knowledge** on the part of the designer;

Recap- continued

- The general form of a design method, and its **three major components**: the representation part, the process part, and the heuristics;
- How to go about **recording the results** of the design process, presenting an **ideal view** of design development;
- Some of the factors that affect the operation of design teams, and how this differs from individual design practices.

Overview

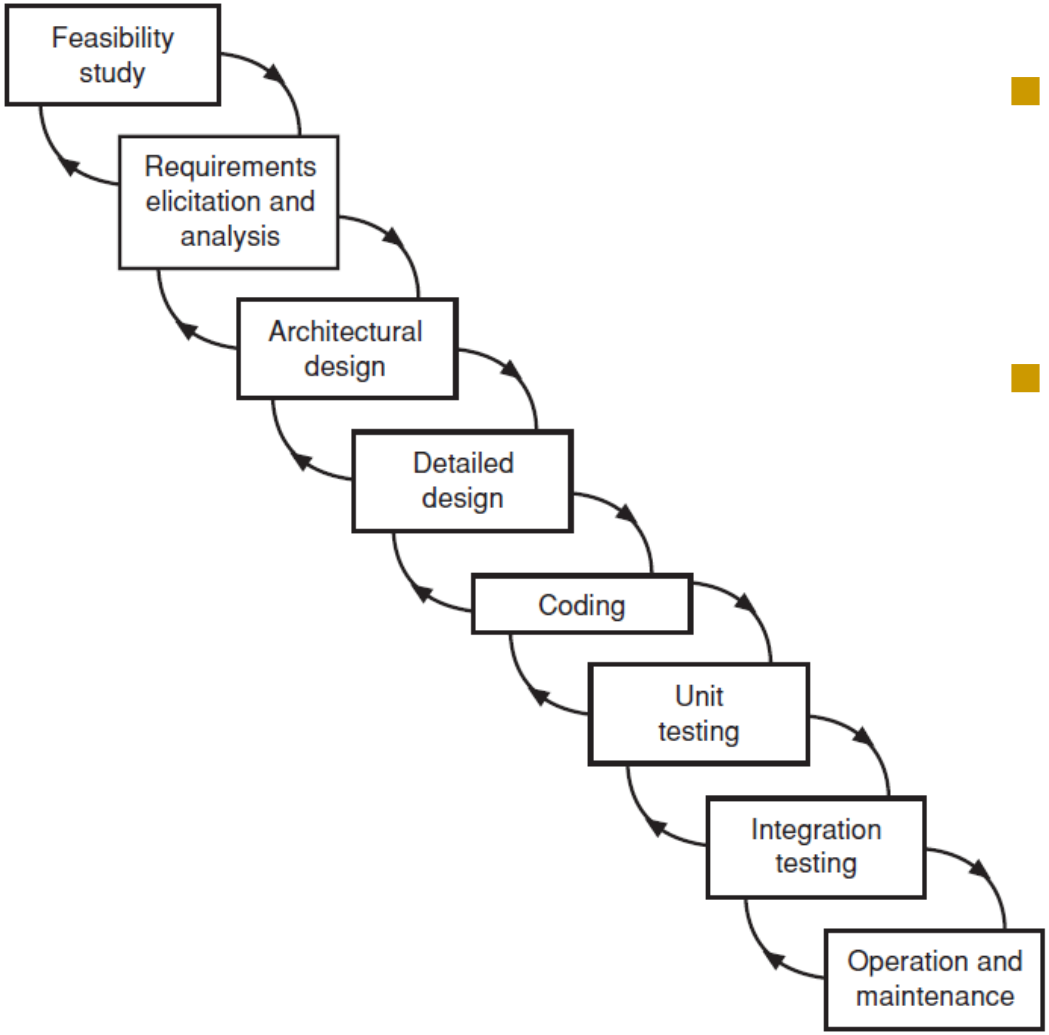
- A context for design
- Linear development processes
- Incremental development processes
- Economic factors
- The longer term

A context for design

- ‘Programming in the large’
- ‘Software life-cycle’

- Examples of Software Development Processes
  - Linear
  - Incremental
  - Reactive

Linear development



- Waterfall model (Royce, 1970) is a good example...
- Addresses two questions:
  1. What should we do next?
  2. How long should we continue to do it?

Activity

- Discuss the advantages and disadvantages of the waterfall model.
- Try to find the main benefit and the main risk for every step.

Mental models

- Users develop an understanding of a system through learning about and using it
  - Internal constructions of some aspect of the external world
- Knowledge is sometimes described as a mental model:
  - How to use the system (what to do next)
  - What to do with unfamiliar systems or unexpected situations (how the system works)
- People make inferences using mental models of how to carry out tasks
- Deep versus shallow models
  - e.g. how to drive a car and how it works

Example - Discuss

- A computer system that is to support the issue, reservation and recall of books, CDs and videos for the university library.

- *Hint: The specification for this system may also need to describe the way in which the users' interactions with the final system are to be organized (the 'look and feel' aspects). The interface should provide the librarians with a structure that matches their own* ***mental models*** *of the processes that are involved.*

Example

- Did we manage to specify the ‘real’ needs of the users?
- In which domains is this particularly difficult?

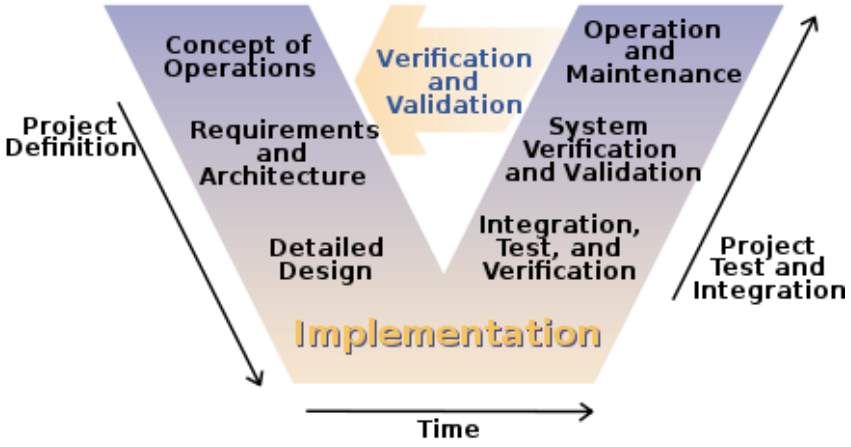
Linear development processes

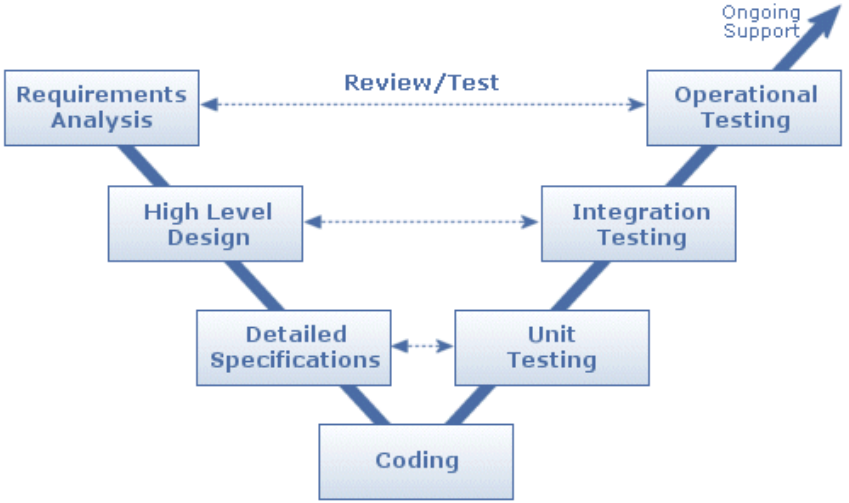
Development process	Reasons for adoption	Typical domains
Linear (e.g. 'waterfall')	Where the needs are reasonably well determined and a 'longer' delivery time is acceptable.	Many 'made to measure' systems are developed in this way. This approach is one that has been used very widely and is still appropriate for systems that need to demonstrate any form of high integrity.
Incremental	Where there may be a need to demonstrate feasibility of an idea; establish a market position rapidly; or explore whether a market might exist. Providing a rapid time to market is often an important factor for adopting such an approach.	Widely used for 'shrink wrap' software of all sorts, such as office packages, games, operating systems. Also, where there is a need to interact closely with the end-users during development.
Reactive	Where evolution of a system is largely in response to what the developers perceive as being needed.	Open source software of all kinds is generally developed in this way, as are many websites.

Linear development processes

- **The waterfall model:**
  - Strong management framework
  - You can define milestones
  - There is a «general plan» (a good thing?)
  - Better to describe progress, as opposed to charting new territory
  - Attractive for buyers (stable budgeting)
- **The transform model:**
  - Convert a formal specification into a program.

V Model





Clarus Concept of Operations. Publication No. FHWA-JPO-05-072, Federal Highway Administration (FHWA), 2005

http://harmonicss.co.uk/project/the-death-of-the-v-model/

Incremental development processes

Development process	Reasons for adoption	Typical domains
Linear (e.g. 'waterfall')	Where the needs are reasonably well determined and a 'longer' delivery time is acceptable.	Many 'made to measure' systems are developed in this way. This approach is one that has been used very widely and is still appropriate for systems that need to demonstrate any form of high integrity.
Incremental	Where there may be a need to demonstrate feasibility of an idea; establish a market position rapidly; or explore whether a market might exist. Providing a rapid time to market is often an important factor for adopting such an approach.	Widely used for 'shrink wrap' software of all sorts, such as office packages, games, operating systems. Also, where there is a need to interact closely with the end-users during development.
Reactive	Where evolution of a system is largely in response to what the developers perceive as being needed.	Open source software of all kinds is generally developed in this way, as are many websites.

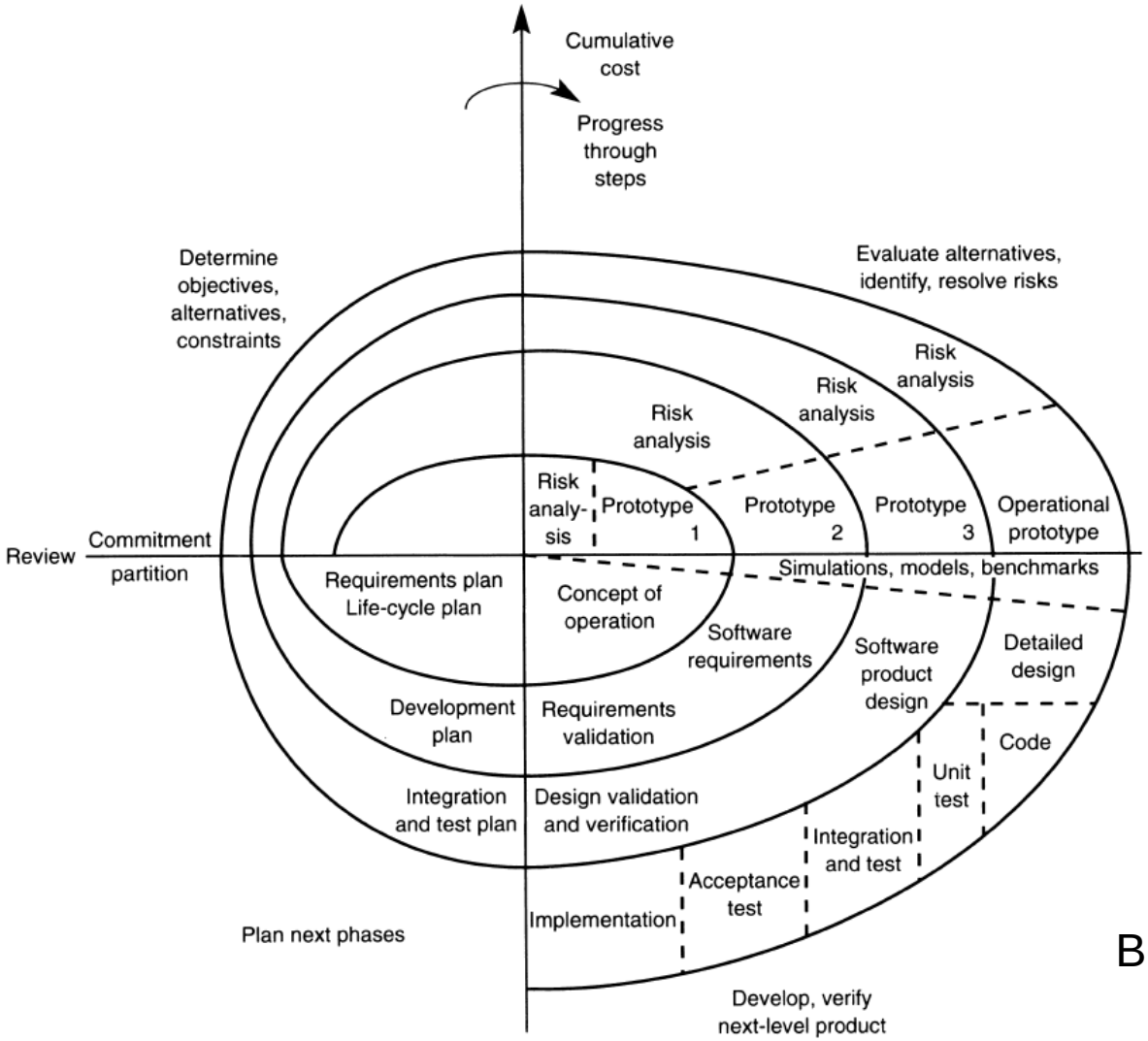
Incremental development processes

- When requirements cannot be determined precisely from the start
- Prototypes may help, as they can be... (Floyd, 84):
  - Evolutionary: for incremental design
  - Experimental: to evaluate a possible solution
  - Exploratory: to clarify user requirements

Prototyping

- What is a prototype?
- Why prototype?
- Different kinds of prototyping
  - low fidelity vs. high fidelity
- Construction
  - evolutionary vs. throw-away

Spiral model of software life-cycle

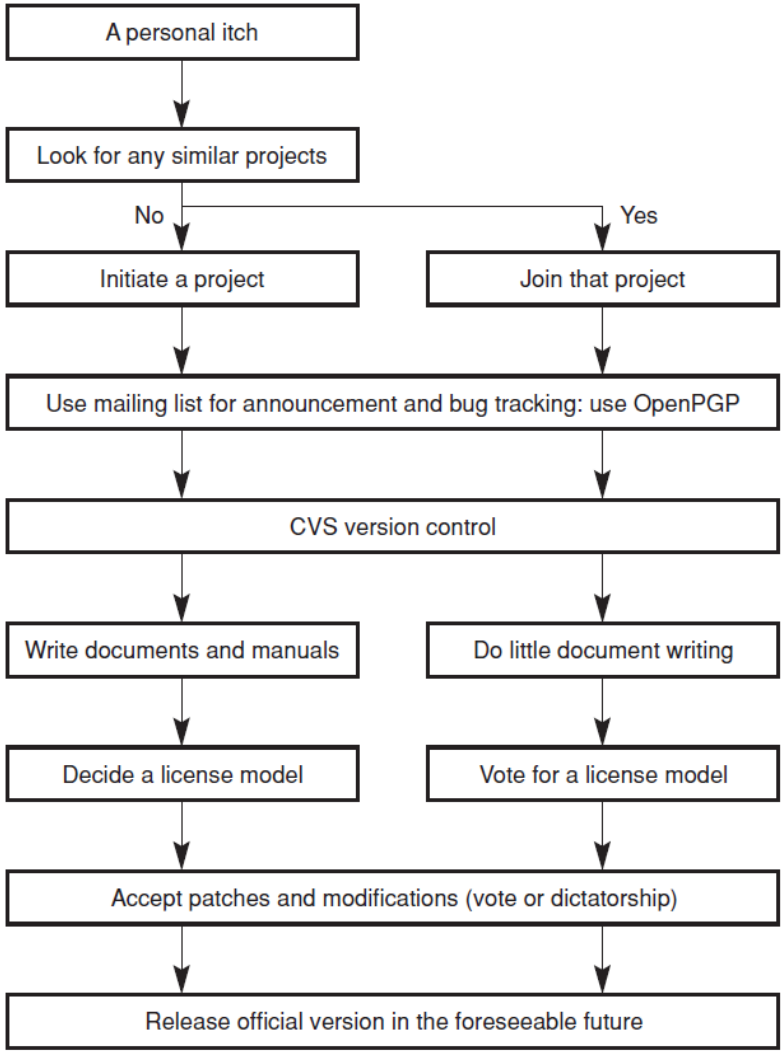


Boehm, 1981

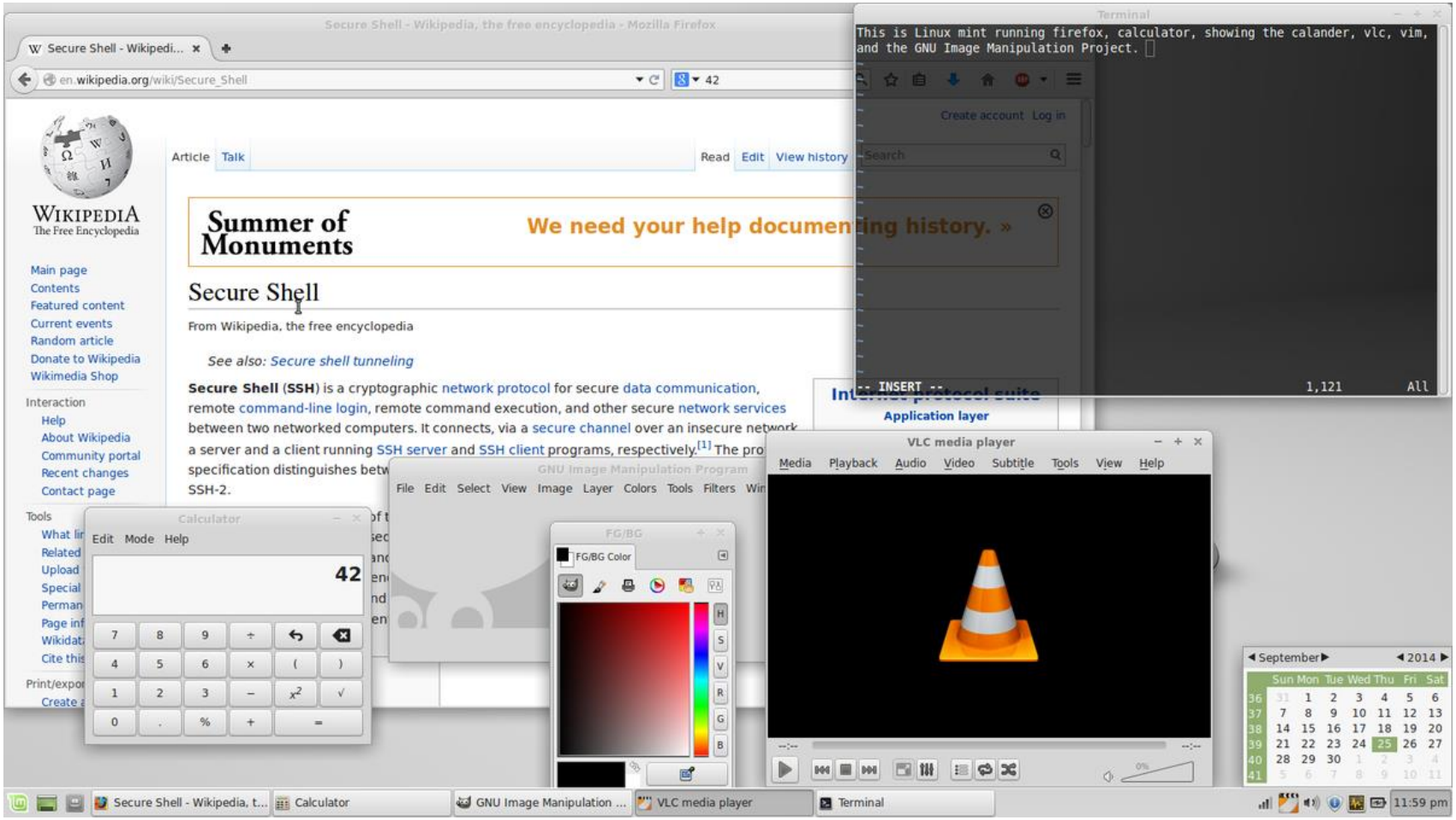
Reactive development processes

Development process	Reasons for adoption	Typical domains
Linear (e.g. 'waterfall')	Where the needs are reasonably well determined and a 'longer' delivery time is acceptable.	Many 'made to measure' systems are developed in this way. This approach is one that has been used very widely and is still appropriate for systems that need to demonstrate any form of high integrity.
Incremental	Where there may be a need to demonstrate feasibility of an idea; establish a market position rapidly; or explore whether a market might exist. Providing a rapid time to market is often an important factor for adopting such an approach.	Widely used for 'shrink wrap' software of all sorts, such as office packages, games, operating systems. Also, where there is a need to interact closely with the end-users during development.
Reactive	Where evolution of a system is largely in response to what the developers perceive as being needed.	Open source software of all kinds is generally developed in this way, as are many websites.

Open source software development



Open source software



"Desktop-Linux-Mint" by Benjamintf1 - Own work. Licensed under CC BY-SA 4.0 via Commons - https://commons.wikimedia.org/wiki/File:Desktop-Linux-Mint.png#/media/File:Desktop-Linux-Mint.png

Open source software

- Richard Stallman announces GNU in 1983; users are free to:
  - run the software,
  - share it (copy, distribute),
  - study it and
  - modify it
- GNU development initiated in 1984.
- Stallman starts Free Software Foundation in 1985.
- GNU was developed and available in 1992.
- Raymond and Perens form Open Source Initiative in 1998.

img. by Etienne Suvasa

Reactive development processes

- Website development
  - Focus on co-ordination of style
  - Growing use of active elements
  - Script based (evolutionary) development
  - *Web engineering* is used as a term

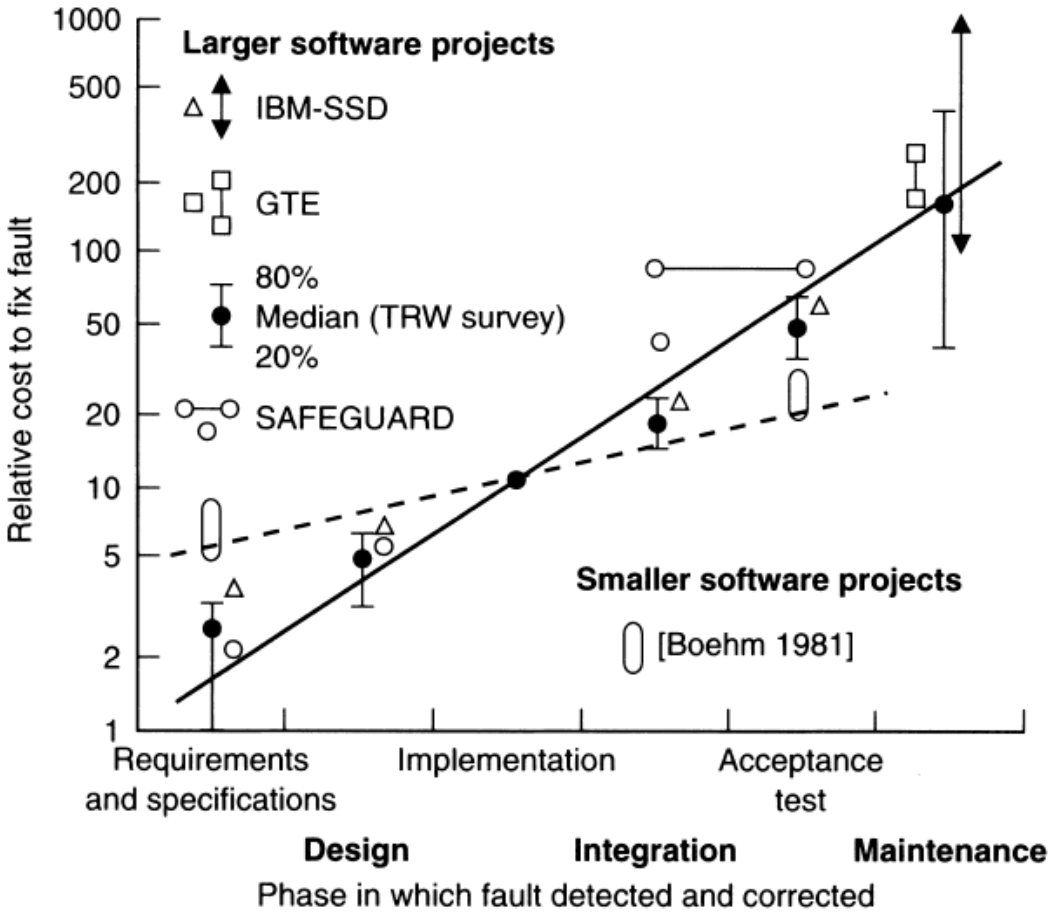
Economic factors

- Software design is not analytic (there is no easily bounded solution space to explore!)
- There may be errors:
  - the design is not self-consistent
  - the design and the specification are not consistent
  - the design and the requirements are not consistent

Economic factors

- Finding inconsistencies:
  - when the design is being refined,
  - while the design is being implemented
  - while the implementation of the system is being tested
- Use design reviews, unit tests, and full tests.
- Boehm's taxonomy of V & V:
  - *Verification*: are we building the product right?
  - *Validation*: are we building the right product?

Cost of fixing bugs



Cost of making changes

- Three forms of maintenance according to Lientz and Swanson (1980):
  - *Perfective* maintenance is concerned with extending and improving a system.
  - *Adaptive* maintenance is performed in order to meet needs arising from external factors.
  - *Corrective* maintenance is performed to fix any ‘bugs’.
- 65 per cent of maintenance is of the first (perfective) category.

The longer term

- Perfective maintenance vs. Evolution
- Example: The Y2K problem
- Lehman and Ramil's classification (2002):
  - **E-type**
    - A system is one that is used to 'mechanise a human or societal activity'
  - **S-type**
    - one which is only required to meet a 'formal' specification.

Summary

- We looked at:
  - design in the context of particular life-cycle models, most notably the linear (waterfall) and incremental models;
  - the associated role of requirements elicitation in setting the context for the various design activities;
  - the roles that software prototyping can play in supporting aspects of the design process;
  - the question of how well a design meets the users' needs, and the roles of:
    - verification: are we building the product right?
    - validation: are we building the right product?
  - the cost of fixing errors in design at a later stage of development;
  - the influence of the *maintenance* phase

Further reading

- Boehm B.W. (1988). A spiral model of software development and enhancement. *IEEE Computer*, **21**(5), 61–72
- Lehman M.M. and Ramil J.F. (2002). Rules and tools for software evolution planning and management. *Annals of Software Eng.*, **11**(1), 15–44

Questions?

Exercise

- List reasons why each of the following systems may require ‘maintenance’ in the future. Identify the relevant form of maintenance likely to be employed for each case:
  a) a spell-checking program used with a word processor that supports multiple languages;
  b) a program used to list the contents of a directory (folder) used in an operating system;
  c) a system used to display information about the position of aircraft in an air traffic control system that is used to manage the airspace for a number of busy airports.