CSCI222 System Development

Introduction to Software Engineering and System Development

Software Engineering

- Engineering vs. Science
- Software Engineering is

  "the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software" (IEEE standard 610.12-1990).

Components of Software Engineering

2 main components –

- PRODUCT
  - The actual software product or system that is built and put into operation
- PROCESS
  - A framework for the tasks that are required to build high-quality software.

Now we look at …

- Software Engineering Development Activities
- Planning
- Requirements
- Design
- Verification and validation
- Maintenance and evolution

What is Engineering?

- A body of knowledge used when building things
  - Scheduling
  - Costing
  - Estimating
  - Building
  - Testing
  - Communicating
  - Organising

It is easy to build something if you have unlimited money and time. A professional differs from an amateur in that they can contain costs and time.

How software is different?

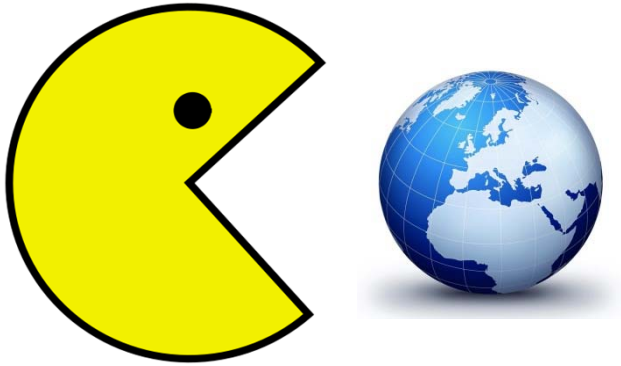
- Software is soft and intangible
- There are no physical laws underlying software behaviour
- Software are never wears out
  - traditional reliability measures don't apply
- Software is not mass produced
- The specification for software continuously changes

"Software eats the World"

- "*We are in the middle of a dramatic and broad technological and economic shift in which software companies are poised to take over large swathes of the economy*"

  (Marc Andreessen, "Why Software is Eating the World", **The Wall Street Journal**, http://online.wsj.com/article/SB10001424053111903480904576512250915629460.html)

- More and more major businesses and industries are being run on software and delivered online services.



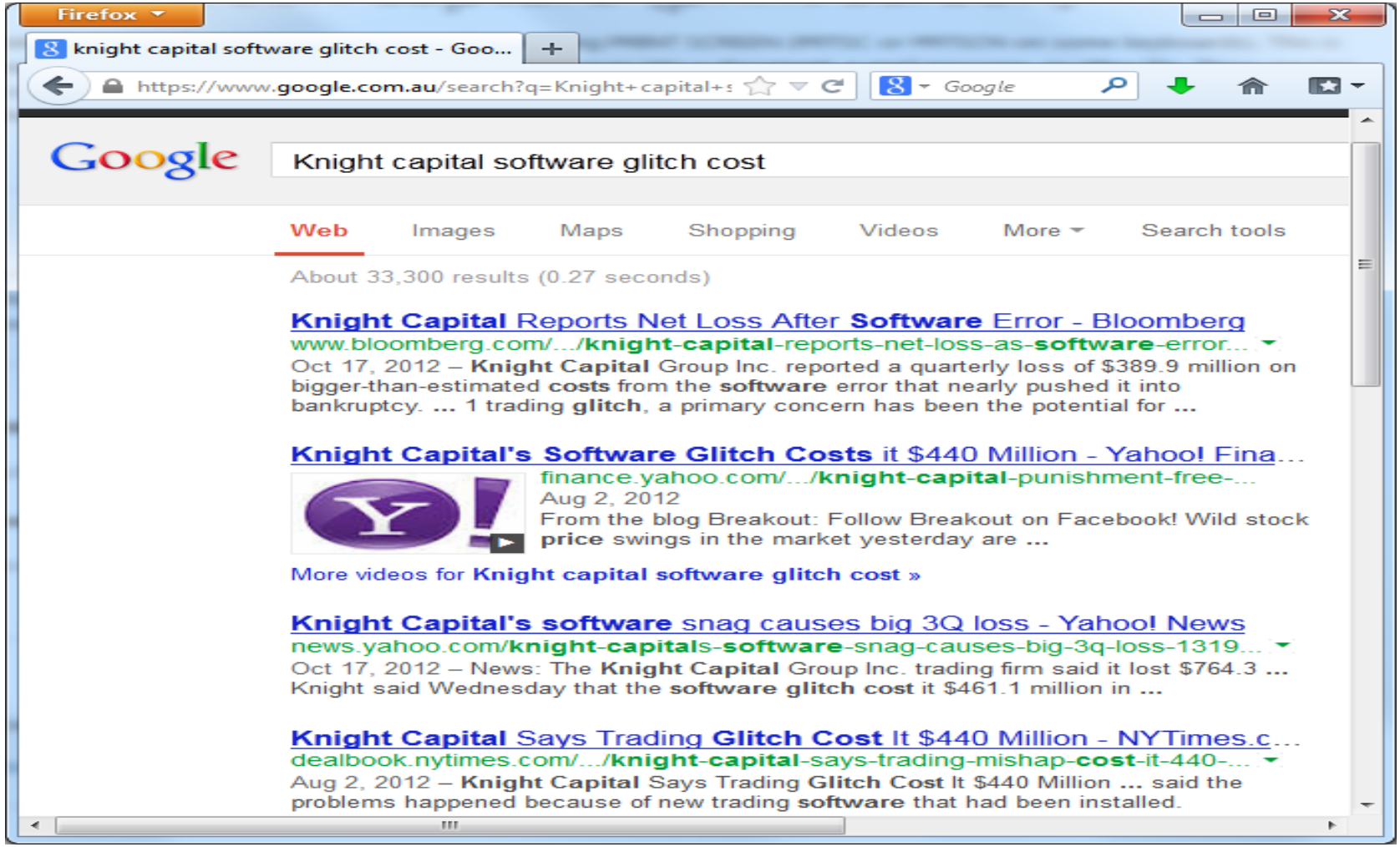
“Software eats the World” (cont.)

Good news for us ….

But “*With great power there must also come great responsibility*” …

Low-quality software costs jobs ...

https://www.youtube.com/watch?v=7BKNnpJfWII



Low-quality software costs money ...

On the 4th June 1996 at 1233 GMT (UTC) the European Space Agency launched a new rocket, **Ariane 5**, on its maiden unmanned flight,

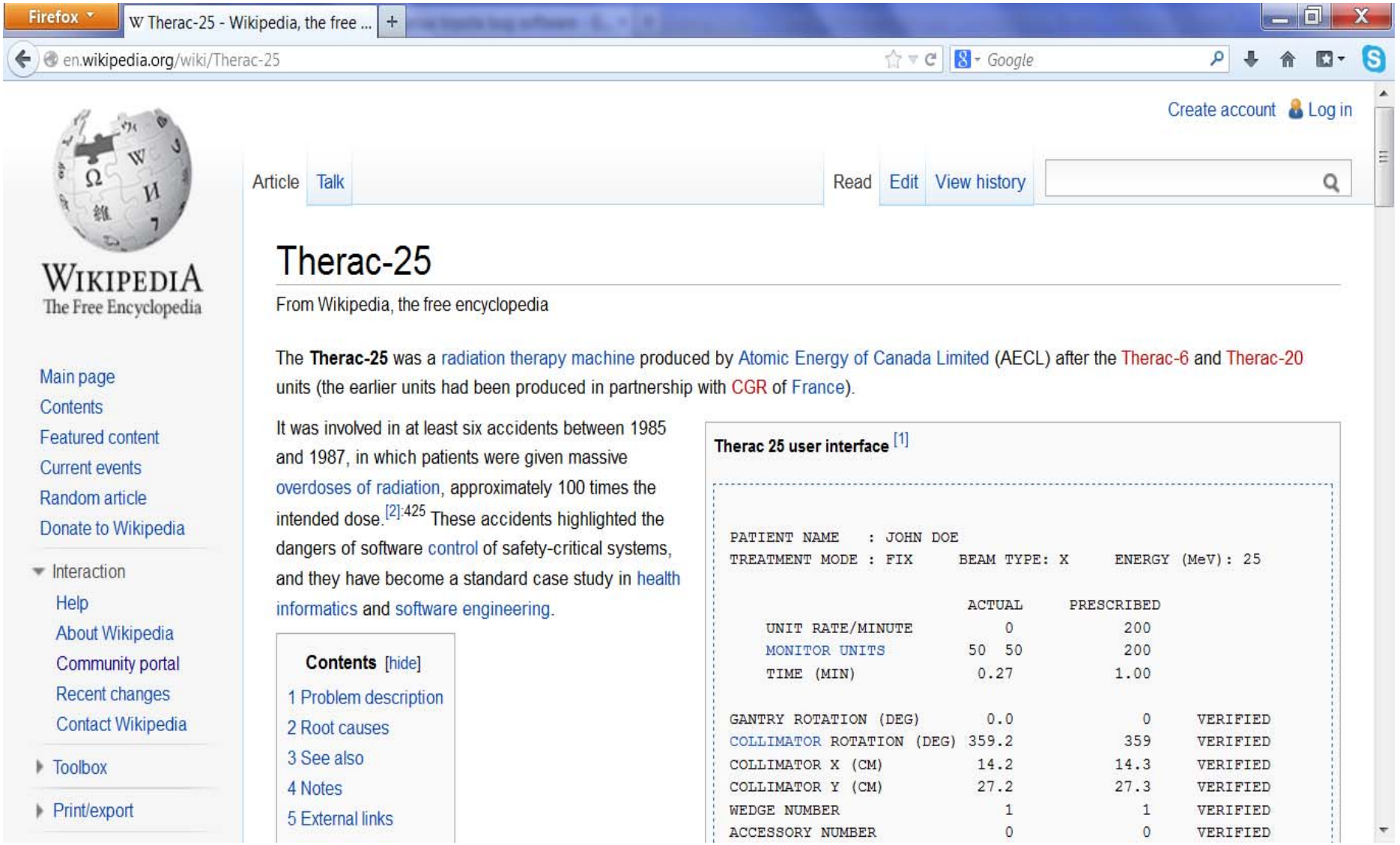
https://www.youtube.com/watch?v=kYUrqdUyEpI

"It took the European Space Agency **10 years** and **$7 billion** to produce **Ariane 5**, a giant rocket capable of hurling a pair of three-ton satellites into orbit with each launch and intended to give Europe overwhelming supremacy in the commercial space business.

All it took to explode that rocket less than a minute into its maiden voyage last June, scattering fiery rubble across the mangrove swamps of French Guiana, was a **small computer program** trying to stuff a 64-bit number into a 16-bit space"

Source: http://www.around.com/ariane.html

Low-quality software costs lives



Software

The Airbus A380 uses a substantial amount of software to create a "paperless" cockpit

Q: How many lines of code constituting the plane's software?

A: **Millions** of lines of code

Android OS has 15 million lines of code. Moodle have 1 million lines of code. Windows XP has 45 million lines of code.

Software Engineering plays a critical part in the development of such a large-scale software.

Question of the day

- If there is a software that you dream of building, what would it be?

Definitions: Systems Development

- System:
  - A system is an integrated composite of *people*, *products*, and *processes* that provide a capability to satisfy a stated need or objective.

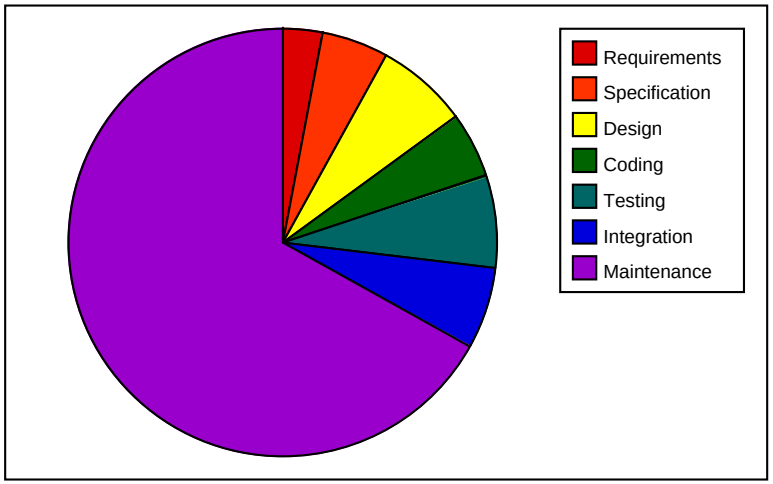
- Systems Development:
  - Process that includes
    - *identifying* information needs,
    - *designing* information systems that meet those needs, and
    - *putting* those systems into practical operation

Systems development

- Technical issues
  - Coding rules
    - Details of actual code … class design … "physical organization" of code
  - Approaches to testing
  - Source code management
  - Deployment
  - Iterative development
  - Identification of classes, use of patterns and frameworks
  - Design models
  - High level design
  - Use cases
- "Human factors"
  - Collaboration within a team
  - Roles and skill sets
  - Time management
  - …
- Communications
  - A common language for technical developers and non-technical users?
  - …

There's more to software than the code

- CSCI114, CSCI124, CSI204 – *you learn to write code*
- Coding as a part of "software development"?
  - Typical estimate says coding represents about 10% of the work.
- Most of the time devoted to other activities
  - Discussions with potential users that determine the functionality of system that will be developed
  - Designing its main components (possibly different processes on different machines)
  - Sorting out "use cases"
  - Interactions with users during refinement cycles
  - …



Software Engineering Development Activities

- Planning
- Requirements analysis
- Design
- Implementation
- Verification & Validation
- Maintenance and evolution

Planning

- Identify business value
- Analyse feasibility
- Develop work plan
- Staff the project
- Estimate
- Identify risk

Planning
Identify business value

- Develop System Request (SR) to initiate the project
  - identifies the business value of new system
  - see example of CD Selections
- An SR usually includes:
  - business need (e.g. to increase CD sales by reaching new (internet) market)
  - functionality – very high level only
  - expected value ($$$$$$) e.g. expected increase in sales
  - other issues/constraints e.g. to be more competitive
- SR will get approved by senior executives / managers (approval committee)

Planning
Analyse feasibility

If expected value and business need are approved by S.R. committee, a **Feasibility Study** is conducted, usually by a Business Analyst.

- **Feasibility Study** – asks "what are the consequences if we invest?"
  - Technical: can we build it?
  - Economic: will it provide business value?
  - Organizational: if it builds, will it be used?

Technical Feasibility

- Can it be built with the technology? (if not, then it means more risk)
- Are we familiar with the technology? (if not, then it means more risk)
- Are we familiar with this type of system? (if not, then it means more risk)
- Roughly how big is it? (bigger means more risk)

Economic Feasibility

- Or Cost vs Benefit analysis
  - Should the system be built?
  - Calculate the financial benefits and costs
  - Do cost/benefit analysis
    - Development costs, Operation cost
    - Tangible benefits,
    - Intangible or hidden costs, benefits
  - Calculate Return on Investment (ROI)

Organizational Feasibility

- Will the system be accepted by the users and integrated into the organization?
- Conduct stakeholder analysis (someone who is affected by the system – end user, sponsor, management, IS dept)

Planning
Develop work plan

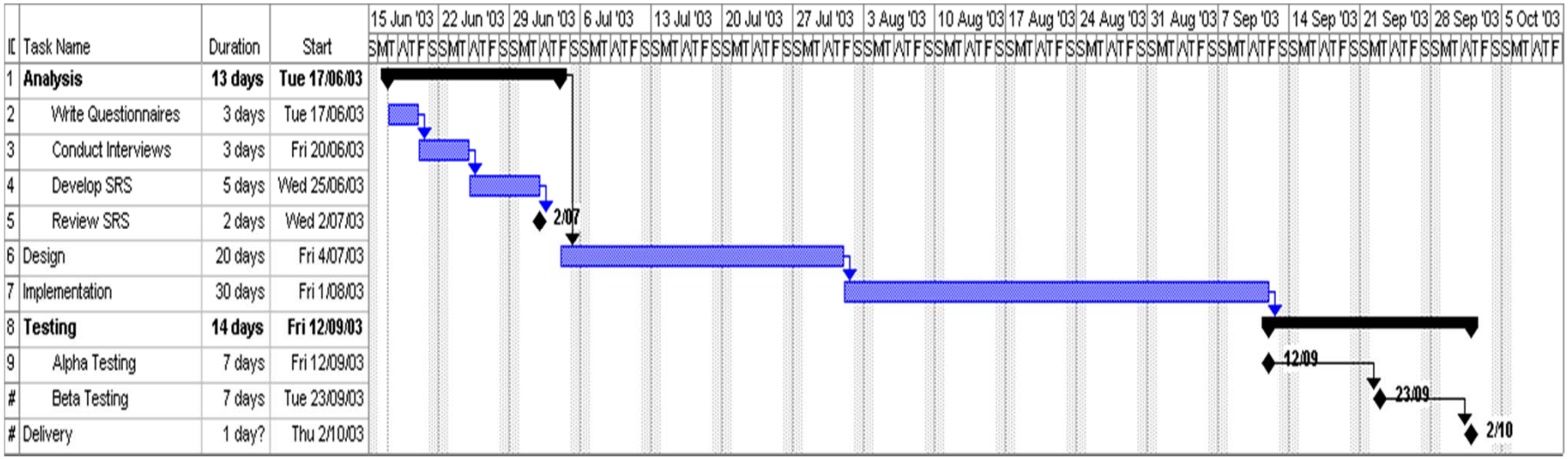
Work Breakdown Structure – WBS

- Identify tasks and subtasks needed to build the system
- Estimate time taken to complete tasks
- Decide what the deliverable will be for that task
- Assign tasks to team members

Example WBS:

TASK	TIME	STAFF	DELIVERABLE
1. Analysis			
1.1 Write Questionnaires	3 days	John	Manager and user Questionnaire
1.2 Conduct Interviews	3 days	Stephen, John	Interview report
1.3 Develop SRS	5 days	Stephen, John	SRS document draft
1.4 Review SRS	2 days	Stephen, John, Jane, Gill, Barry (customer)	Reviewed and approved SRS

GANTT Chart



ID	Task Name	Duration	Start
1	**Analysis**	**13 days**	**Tue 17/06/03**
2	Write Questionnaires	3 days	Tue 17/06/03
3	Conduct Interviews	3 days	Fri 20/06/03
4	Develop SRS	5 days	Wed 25/06/03
5	Review SRS	2 days	Wed 2/07/03
6	Design	20 days	Fri 4/07/03
7	Implementation	30 days	Fri 1/08/03
8	**Testing**	**14 days**	**Fri 12/09/03**
9	Alpha Testing	7 days	Fri 12/09/03
#	Beta Testing	7 days	Tue 23/09/03
#	Delivery	1 day?	Thu 2/10/03

Gantt chart shows tasks, sub-tasks and their start date/end date, milestones, task dependencies

Milestone – a date for delivering something

Quiz: true or false?

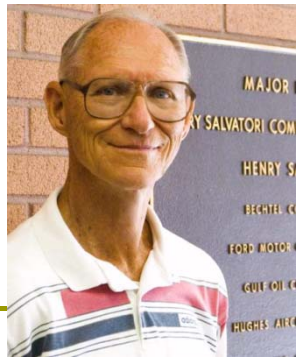
- Software is a product and can be manufactured using the same technologies used for other engineering artefacts.

Planning
Estimate

Estimating time the simple way!

- For "typical" business application systems, calculate the effort it takes for the Planning phase and use industry standard % to calculate the rest of the SDLC phases
  - Planning - 15% effort (person months)
  - Analysis - 20% effort
  - Design - 35% effort
  - Implementation - 30% effort
- Example:
  - Planning has taken 1.5 person months, so…
    - Analysis will take 2.0 person months
    - Design will take 3.5 person months
    - Implementation will take 3.0 person months

Constructive Cost Model (COCOMO)



- Developed by Barry Boehm first in 1981 (COCOMO 81).
  - Latest version is COCOMO II (published in 2000)
- **Basic** model

$$\textbf{effort} = c \times \textbf{size}^{k}$$

  - **c** and **k** depend on the type of system: organic, semi-detached, embedded
  - Size is measured in KLOC (i.e. thousands of lines of code)
  - Effort is measured in person-months

Further source: Barry Boehm et al. *Software estimation with COCOMO II*, Prentice-Hall 2002

The COCOMO constants

System type	c	k
Organic (broadly, information systems)	2.4	1.05
Semi-detached	3.0	1.12
Embedded (broadly, real-time)	3.6	1.20

Effort = c x size^k

- **Organic:** a small team develops a small system with flexible requirements in a highly familiar in-house environment
- **Embedded:** the system has to operate within very tight constraints and changes to the system very costly.
- **Semi-detached:** This combines elements of the organic and the embedded types or has characteristics that came between the two.

Example: what is the estimated effort of developing an organic system with 50,000 lines of code?

Planning
Identify risks

- What is a risk?
  - Something that can impact the ability to deliver the project
    - e.g. supplier cannot supply hardware until one week before project delivery date
- Identify and classify all risks
- Decide on most important – i.e. highest risk
- Prevention or contingency plan

Three strategies for risk reduction

- Avoiding the risk:
  - change requirements, design, process for performance or functionality
- Transferring the risk:
  - transfer responsibility to other system, or buy insurance
- Assuming the risk:
  - accept the risk and find way to mitigate or control the outcome.

Software Engineering Development Activities

- Planning
- Requirements analysis
- Design
- Implementation
- Verification & Validation
- Maintenance and evolution

Requirements analysis

- The process of establishing what services are required and the constraints on the system's operation and development.
- There are two major activities:
  - Requirements elicitation
    - Who are the stakeholders of this project?
    - What do the system stakeholders require or expect from the system?
    - Interviews, questionnaires, meeting, etc.
  - Requirements specification
    - Defining the requirements in detail
    - Use cases are the major part of a requirements specification
    - Artefact: Software Requirements Specification (SRS)

Software Engineering Development Activities

- Planning
- Requirements analysis
- Design
- Implementation
- Verification & Validation
- Maintenance and evolution

Design

- What is Design?
  - Design is concerned with HOW we build a system
- How is Design different from Requirements Analysis?
  - Analysis is concerned with WHAT is to be built
- There are many different aspects to design

Design (cont.)

Design comprises of different aspects:

1. Architectural design
2. Sub system design
3. Detailed design
4. Persistent data design
5. GUI design

Architecture Design

Concerned with major structure of the software

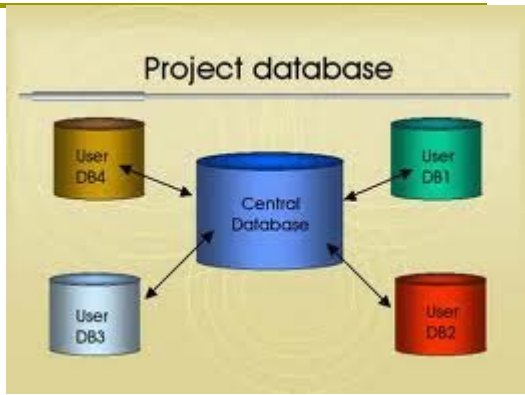
- Similar to building architecture (e.g. roof, walls, foundations)
- Contains sub-systems (air-conditioning unit, water pump)
- Uses external services (sewerage, water, gas, electric)
- The most important part of design – if the architecture is wrong, the house will fall down (or software will fall down!)

Design

- Sub-system design
  - Describes interfaces/protocols of each sub-system
  - Structure of each sub-system
  - Structure can describe how each sub-system sub-divides into packages, components
- Detailed design
  - Describes each class, methods, attributes/data-types

Design (cont.)

- Persistent data design
  - Describes choice of database, tables, primary and foreign keys



- User interface design
  - Describes:
  - how the GUI will look like
  - what tools will be used to build it
  - the structure of the GUI
  - what design guidelines will be followed
  - what style guidelines will be followed

Flashback Quiz

- Software deteriorates rather than wears out because
  - **A)** Software suffers from exposure to hostile environments
  - **B)** Defects are more likely to arise after software has been used often
  - **C)** Multiple change requests introduce errors in component interactions
  - **D)** Software spare parts become harder to order

Software Engineering Development Activities

- Planning
- Requirements analysis
- Design
- Implementation
- Verification & Validation
- Maintenance and evolution

Implementation

- Construction
  - Write the code for the project.
- Installation (or deployment)

You know how to code don't you?

- You will be doing coding in CSCI222, CSCI204, CSCI311, CSCI318, CSCI399, CSCI321, and in your sleep …

Move along folks, nothing to see here…

Software Engineering Development Activities

- Planning
- Requirements analysis
- Design
- Implementation
- Verification & Validation
- Maintenance and evolution

Software verification and validation

- Verification and validation (V & V) is intended to show that:
  - a system conforms to its specification and
  - meets the requirements of the system customer.
- Testing is the most commonly used V & V activity.
- System testing involves **executing** the system with **test cases** that are derived from the specification of the **real data** to be processed by the system.

Verification

- It works!
- Final check through all those **use cases** from the specification document
  - Does the system perform exactly as specified?

and Validation

- Bring in the users and demonstrate

You: It is working. Here let me demonstrate how you would …

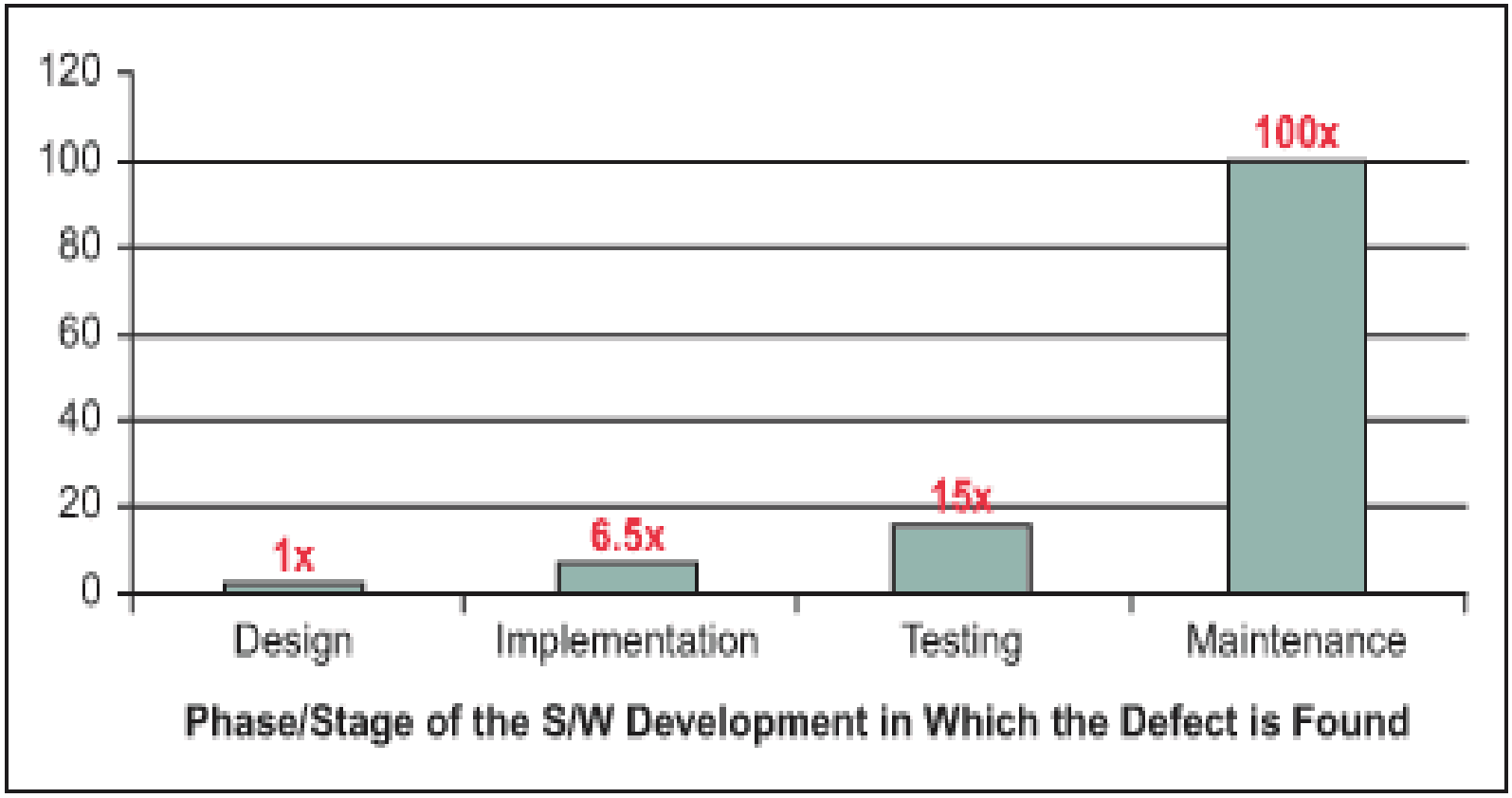
User: That's **not** what I wanted!

I wanted …

You: It is exactly what is in the agreed specification! You said you wanted …

User: You should have implemented what I meant, not what I said.

Cost of defects …



- Cost of correcting an error in requirement specifications increases as we move through lifecycle phases

Software Engineering Development Activities

- Planning
- Requirements analysis
- Design
- Implementation
- Verification & Validation
- Maintenance and evolution

Software maintenance and evolution

- **Manny Lehman**'s law of software evolution:

If a program is any good, then it will have to be changed.

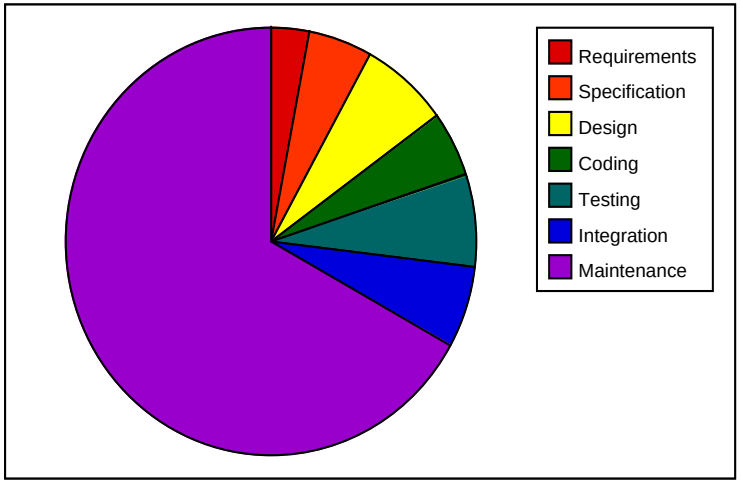
- If the users find they can get work done with the new system, then they will soon identify additional tasks for it to do and require changed forms for interaction with the system.

Types of software maintenance

- Adaptive maintenance
  - Changing the system in response to changes in its environment so it continues to function
- Corrective maintenance
  - Fixing errors & bugs
- Perfective maintenance
  - Changing the system's functionality to meet changing needs

Review

- Planning
- Requirements analysis
- Design
- Implementation
- Verification & Validation
- Maintenance and evolution



Development costs

Flashback quiz

- Which of the following is a good strategy to deal with risks?
  - A. Avoiding the risk
  - B. Transferring the risk
  - C. Ignoring the risk
  - D. Assuming the risk