Dr. Ajitha Rajan

Informatics 2- Software Engineering

Today's Goals

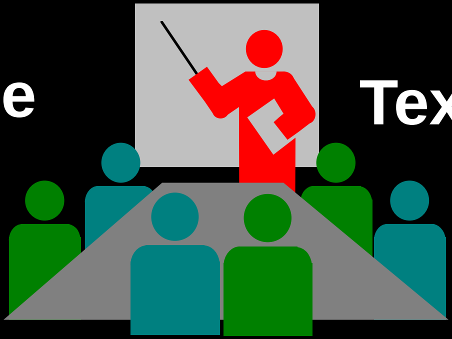
- Understand what Inf2C-SE is all about
  - Instructor and teaching model
  - What you should already know
- Clarify course expectations
  - Coursework
- Answer any questions
- Introduce Software Engineering

Teaching Staff

- Dr. Ajitha Rajan
  - Office: Informatics Forum Room 4.14
  - Email: arajan@staffmail.ed.ac.uk
- TA: Dr. Allan Clark
  - Office: Informatics Forum Room 1.31
  - Email: a.d.clark@ed.ac.uk
- Class information
  - Class web page available at http://www.inf.ed.ac.uk/teaching/courses/inf2c-se/

Learning Modes

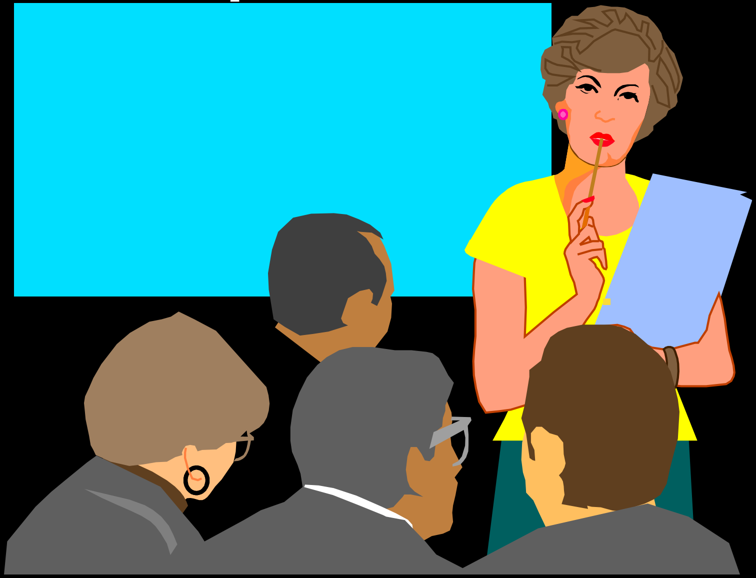
Lecture



Textbooks

Coursework and Lab

Group Discussions



Prerequisites

- Proficient in C++ and Java
  - You should be able to read and write programs without additional instruction
  - This is **not** a programming language class
- Basic understanding of algorithms, logic, and sets

Coursework and Exam

- **Final Exam** worth **60%** of course assessment
  - short answers, written on the question paper

- **Coursework** worth **40%**
  - 3 parts
  - HW1 - Requirements and use cases (20%) (Due 2nd October)
  - HW2 - Design (30%) (Due 16th October)
  - HW3 - Implementation and test (50%) (Due 11th November)

Tutorials and Labs

- There will be 4 tutorials in weeks 2,4,6 and 8.

Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Wk 10

- Tutorials are meant as a platform for discussions relevant to your coursework and any clarifications you may have on Lecture content.

- 12 Tutorial Groups. Check the course web page for tutorial groups, times and location.

- **Lab** will be held in week 7 to help with coursework 3.

Tutorials and Labs

- Note about tutorial in Week 2 only:
- Group 4, normally Old College, will be in Room 5.03, AT
- Group 8, normally Old College, will be in Room 5.07 AT
- Group 9, normally Minto House, will be in Room 5.03 AT
- Group 12, normally 24 Buccleuch Pl., will be in Room 5.03 AT
-
- The rest of the groups will stay in the same rooms throughout.
- Note that the above change is for **week 2 only** and only for **groups 4,8,9 and 12**.
- For the remaining weeks the room displayed on the course webpage is the correct location.

Team Selection

- Coursework will be done in teams of 2 people.
- Select your team and let the TA know by 23rd September by email with the **names and UUN** of the team members. *Only 1 email per team.* The subject of the email should be “Inf2C-SE Team”.

Team Selection

- Coursework will be done in teams of 2 people.
- Select your team and let the TA know by 23rd September by email with the **names and UUN** of the team members. *Only 1 email per team.* The subject of the email should be "Inf2C-SE Team".

Expected Workload

- We do have a sizeable project, this may be a lot of work.
- Planning and scheduling your time is essential
  - Some deliverables may involve quite a bit of work
  - Make sure you spread out the work
  - You will have problems trying to “cram”

Books

- No book is essential.
- The following are worth considering
  - Sommerville, Software Engineering
    - Comprehensive on SE, but limited on UML and Java.
  - Stevens with Pooley, Using UML
    - Covers basic SE, does UML thoroughly, no Java.

News and Schedule

- Check the course web page regularly for any updates and announcements for the course
- Review schedule (in preparation) in the course web page

Questions?

Software Engineering

What is Software engineering?

What is Software Engineering

As defined in IEEE Standard 610.12:
The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

Software Engineering, is also informally defined as, *the branch of computer science that creates practical, cost-effective solutions to computing and information processing problems.*

What is Software Engineering

Building software that works

- We know how to build other things.
- Engineering is the discipline that teaches us the methodologies that work for building complex objects.

Apply engineering techniques to software.

- Know what methodologies work.
- Understand why and how.
- Apply them appropriately and repeatedly.

Software Engineering

- How does software differ from other engineered systems?

Typical Engineered Systems

Engineering Process Model

- Specification
  - Set out the requirements and constraints on the system
- Design
  - Produce a paper model of the system
- Manufacture
  - Build the system
- Test
  - Check the system meets the required specifications
- Install
  - Deliver the system to the customer and ensure it is operational
- Maintain
  - Repair faults in the system as they are discovered

Software is Different

- Set of constraints is continually changing
  - In significant and meaningful ways
  - In unknown and unknowable ways
  - Priorities are changing
- People expect software to adapt
  - More so than anything else
- Software often fails to meet expectations. . .
  - yet we intend to build it anyway
- So how do we go about building software?
  - This is what software engineering is about.

Software Process Models

- Normally, requirements are incomplete and ambiguous and can change during development.
- Very blurred distinction between specification, design and manufacture
- No physical realization of the system for testing
- Software does not wear out
  - Maintenance does not mean component replacement

Software Engineering Myths

- MANAGEMENT

“We have books with rules. Isn’t that everything my people need?”
Which book do you think is perfect for you?

“If we fall behind, we add more programmers”
“Adding people to a late software project, makes it later” – Fred Brooks (The Mythical Man Month)

“We can outsource it”
If you do not know how to manage and control it internally, you will struggle to do this with outsiders

Software Engineering Myths

- CUSTOMER

“We can refine the requirements later”
A recipe for disaster.

“The good thing about software is that we can change it later easily”
As time passes, cost of changes grows rapidly

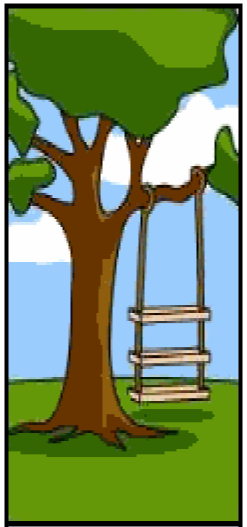
Software Engineering Myths

- PRACTITIONER

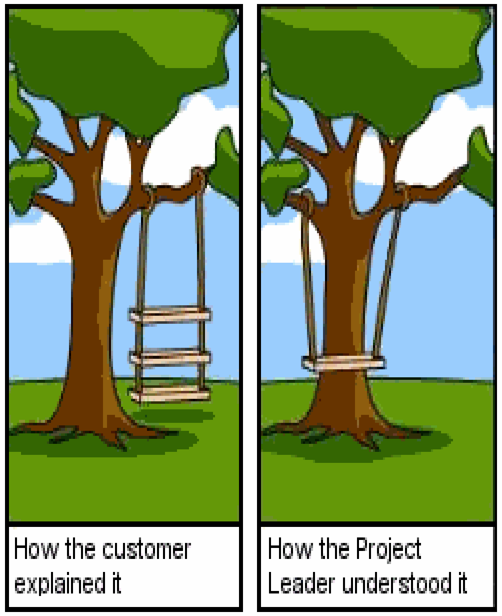
“Let’s write the code, so we’ll be done faster”
The sooner you begin writing code, the longer it’ll take to finish”
60-80% of effort is expended after first delivery

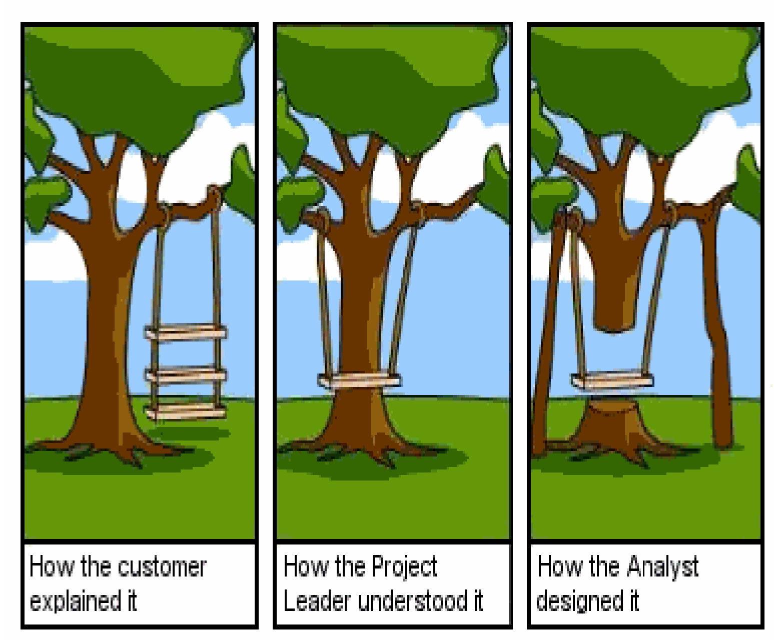
“Until I finish it, I cannot assess its quality”
Software and design reviews are more effective than testing (find 5 times more bugs)

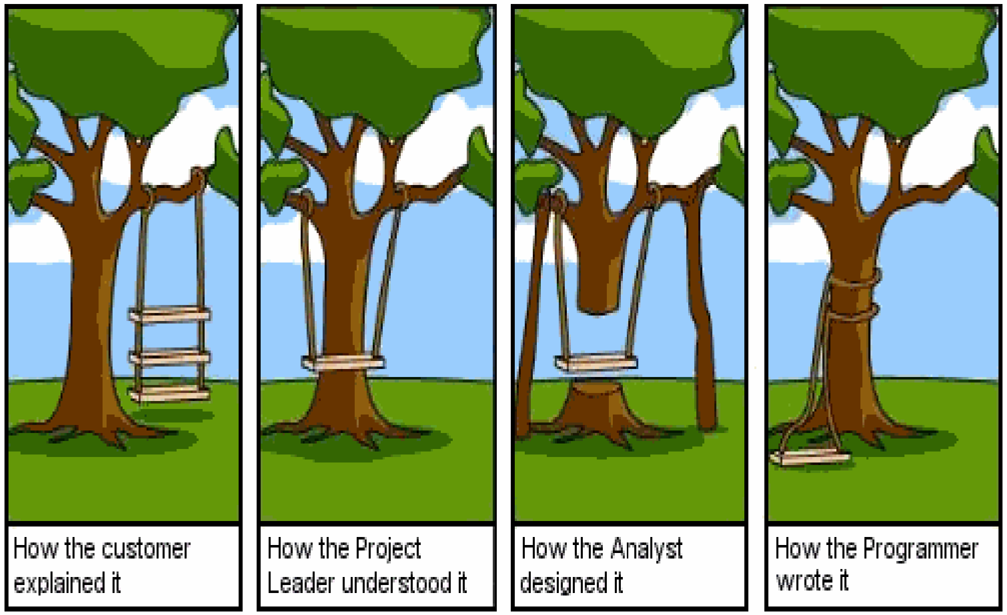
“There is no time for software engineering”
But is there time to redo the software?

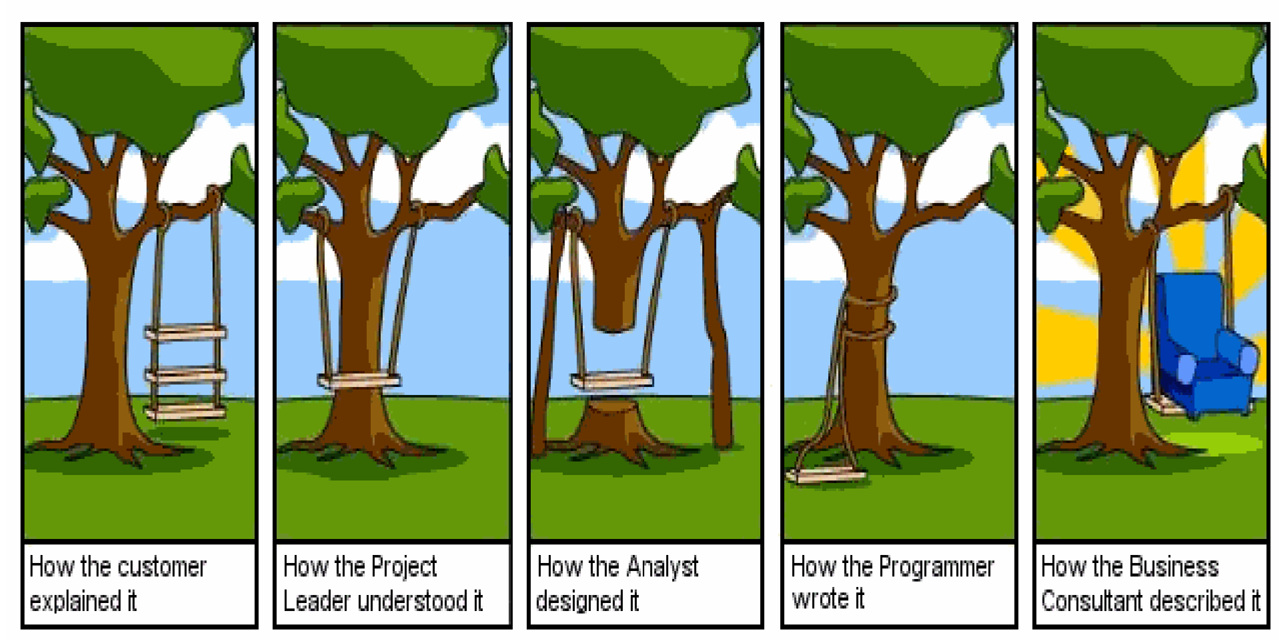


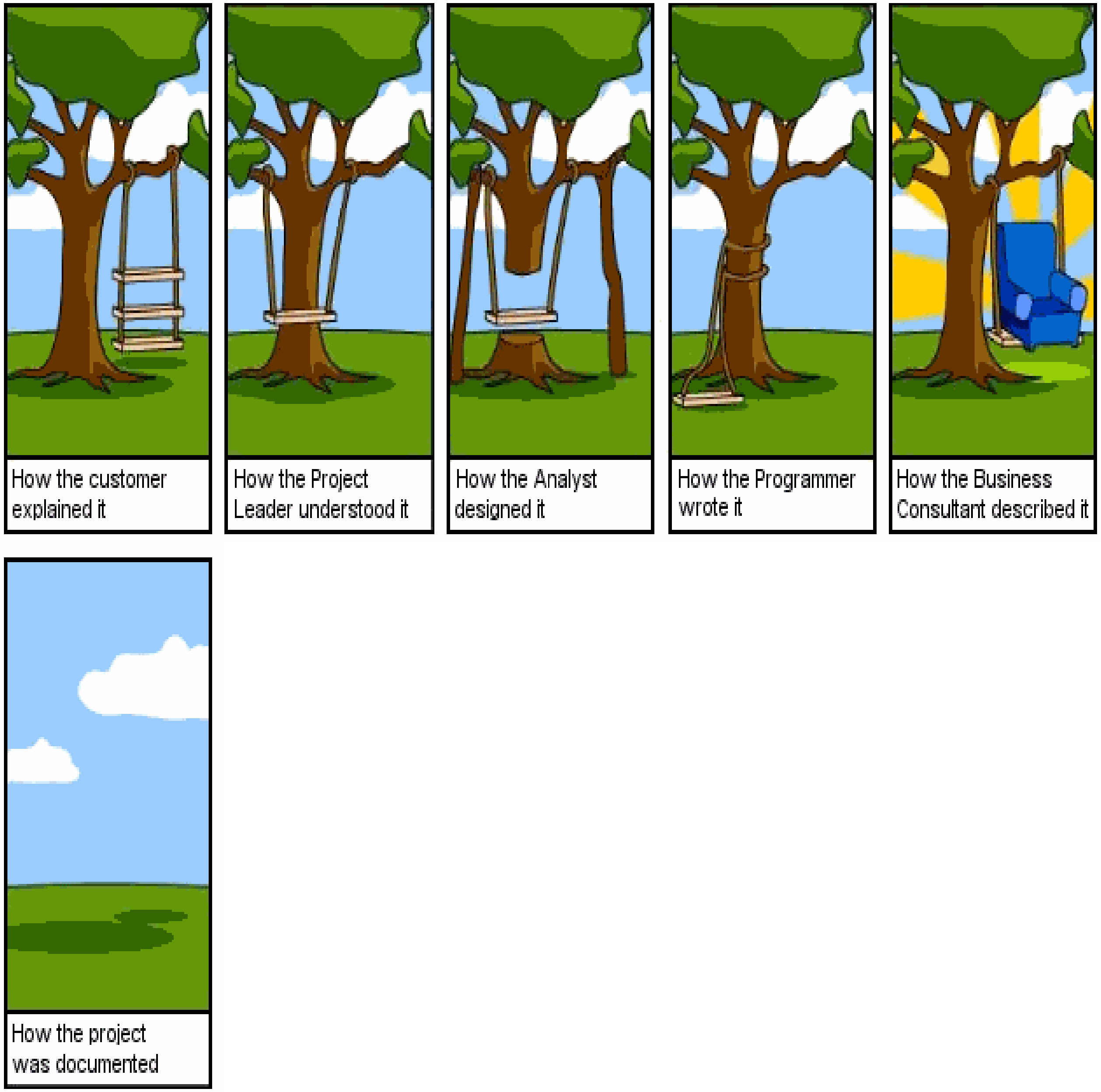
How the customer explained it

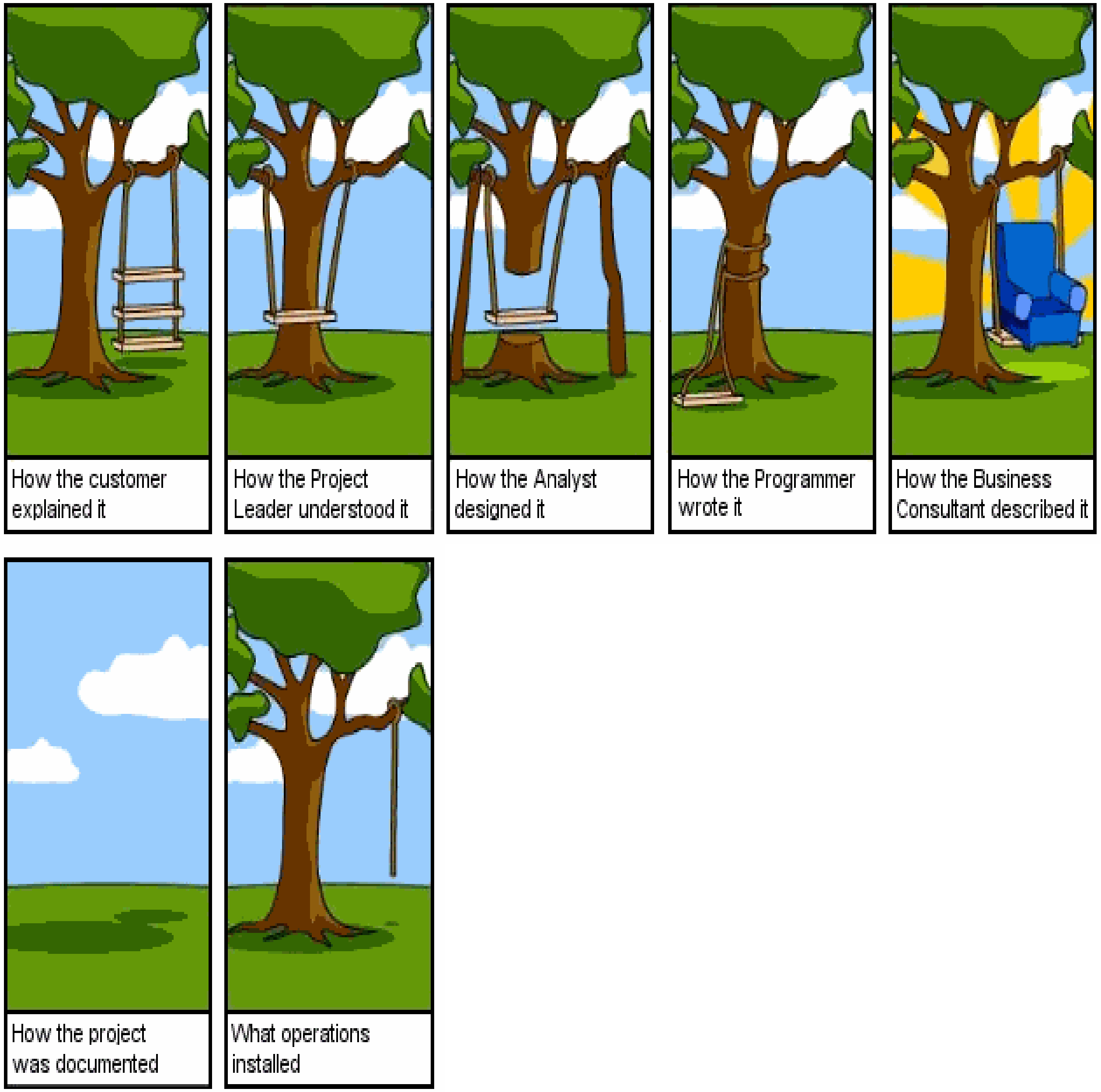

How the customer explained it
How the Project Leader understood it

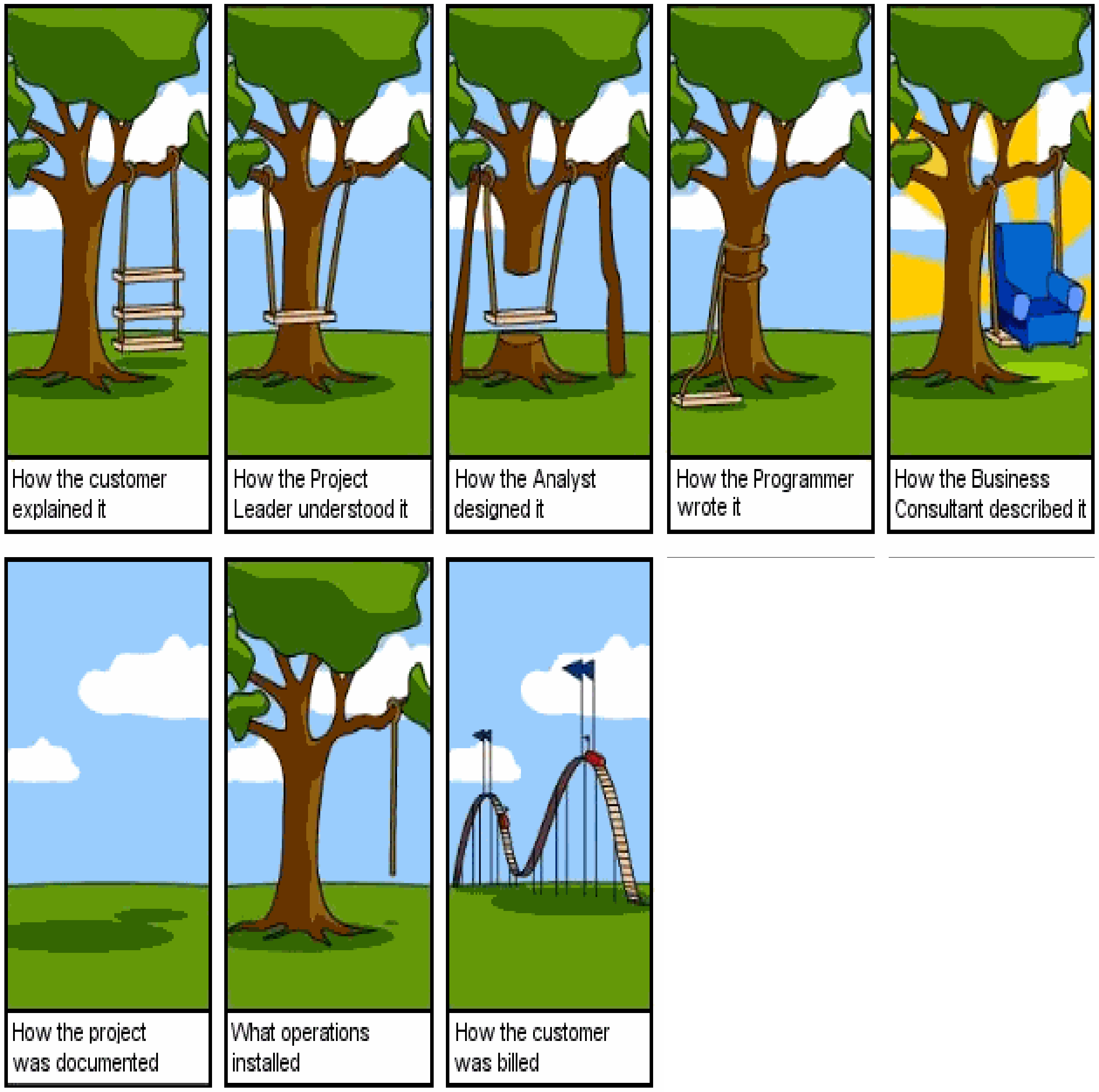

How the customer explained it
How the Project Leader understood it
How the Analyst designed it

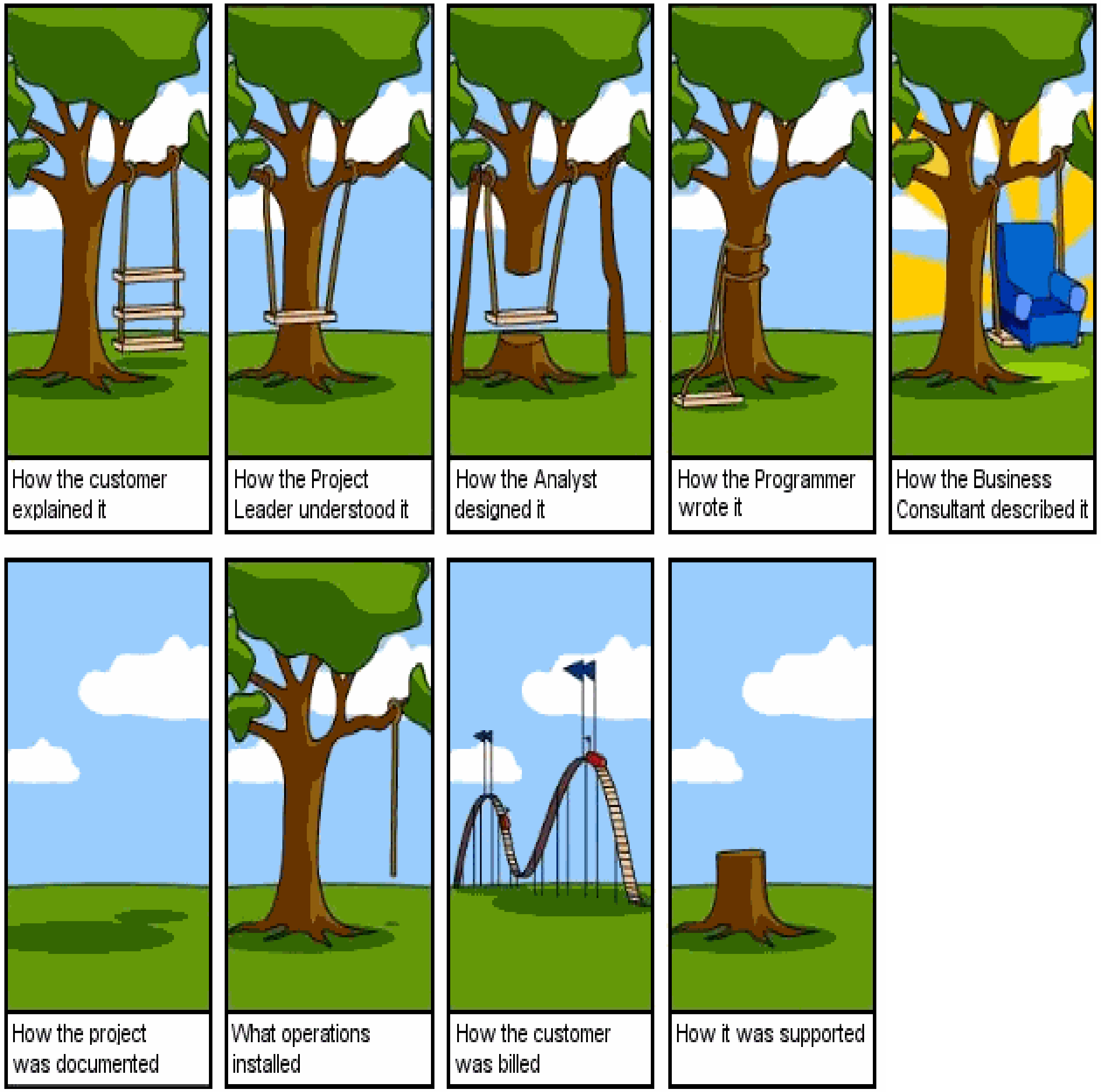

How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it

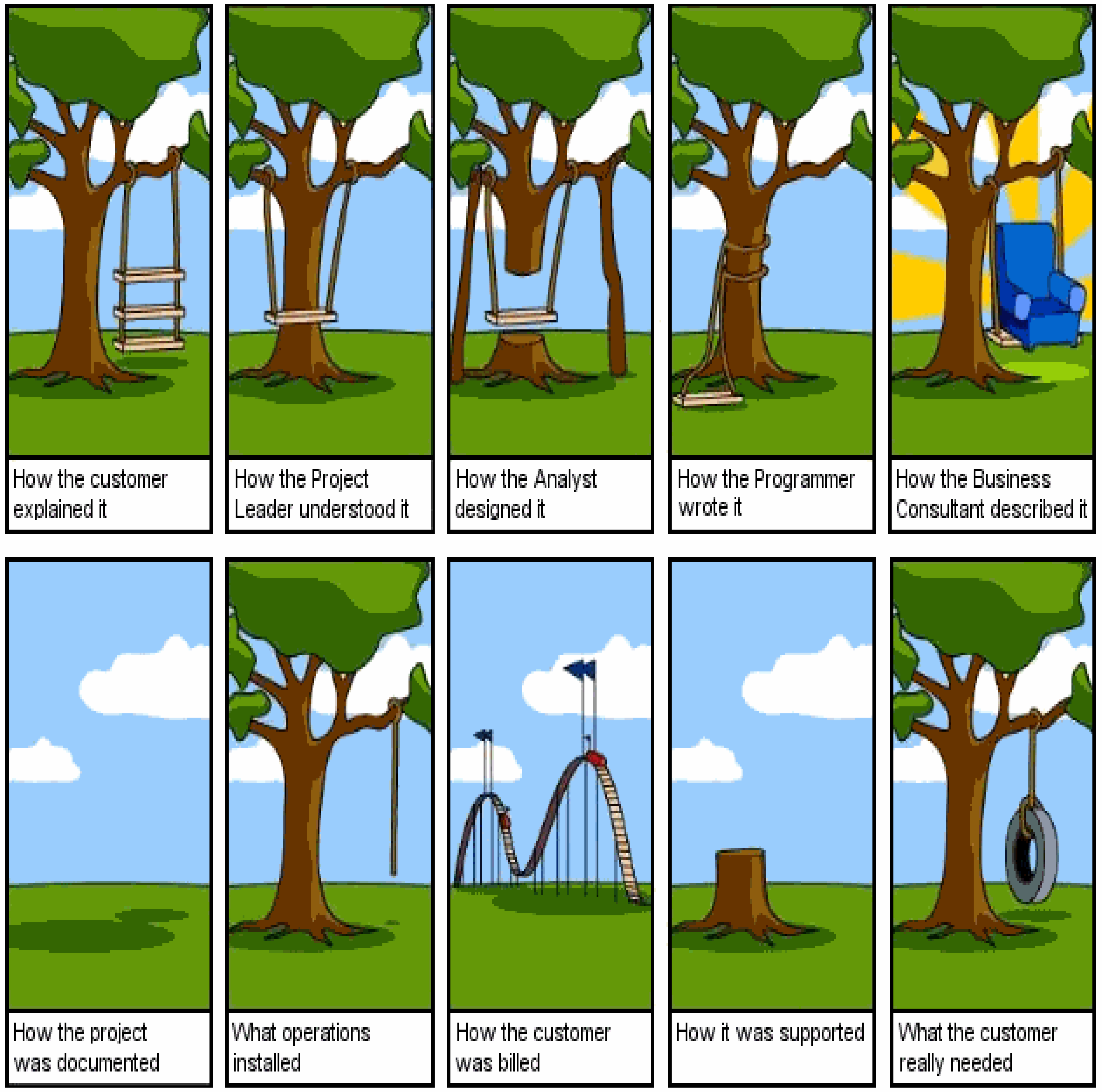

How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Software Engineering

- The economies of **all** developed nations are dependent on software
- More and more systems are software controlled
- Software engineering is concerned with theories, methods and tools for professional software development
- Software engineering expenditure represents a significant fraction of GNP in all developed countries

Software Engineering

Designing, building and maintaining (large) software systems

Software Engineering

What is a Large software?

Examples of Large Software

Mozilla Firefox
12 Million Lines of Code
https://www.openhub.net/p/firefox

Facebook
61 Million Lines of Code

Boeing 787 Flight Software
14 Million Lines of Code

Windows Vista
50 Million Lines of Code

Check out http://www.informationisbeautiful.net/visualizations/million-lines-of-code/
For a visualisation on size of other software codebases

A Question

Are we any good at building software?

The Problems

- Software projects are struggle in the delivery of final product
- Standish chaos reports classify software development projects for medium-large organisations
  - Succeeded
    - 1994:16%...2004: 29%...2009:32%
  - Challenged (i.e., delivered something but maybe reduced scope, late, over budget)
    - No real trend, around 50%
  - Failed (i.e., cancelled without delivering anything)
    - 1994:31%...2004: 18%....2009: 24%

The Haunting of Software Bugs

Recent research at Cambridge University (2013, link) showed that the global cost of software bugs is

around 312 billions of dollars annually

Cyber attacks are affecting nearly anyone that uses computers and causing enormous financial damages

- The Love Bug virus (5/2000) ~$8.7 billion
- Flash crash
- http://www.forbes.com/pictures/fmdk45gmjl/sony/

Software BUGs – SPACE disaster

Maiden flight of the Ariane 5 rocket on the 4th of June 1996

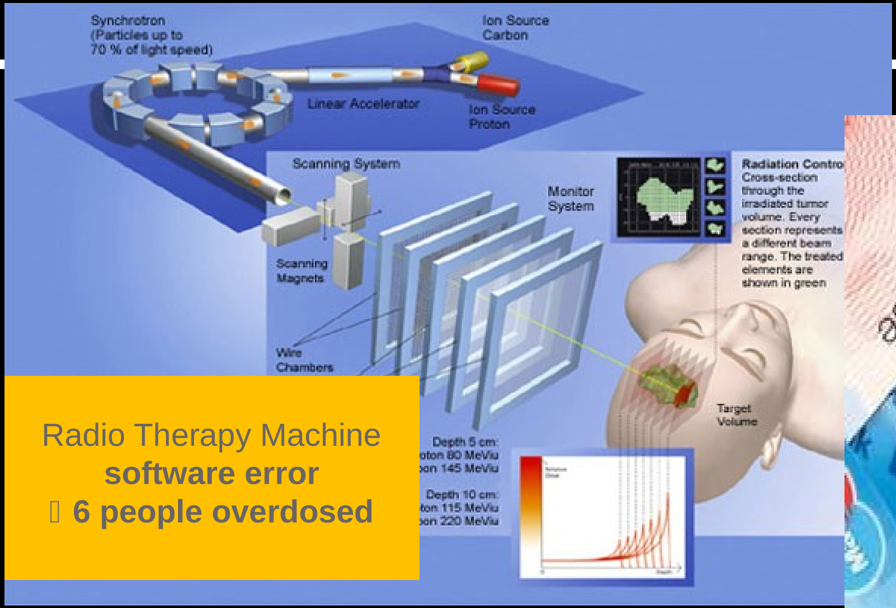
- The reason for the explosion was a software error (Attempt to convert a 64-bit floating point number, representing horizontal velocity, to a 16-bit integer failed)
- Financial loss: $500,000,000
- (including indirect costs: $2,000,000,000)

Boeing could not assemble and integrate the fly-by-wire system until it solved problems with the databus and the flight management software. Solving these problems took more than a year longer than Boeing anticipated. In April, 1995, the FAA certified the 777 as safe.

Total development cost:	$ 3 billion
Software integration and validation cost:	one third of total

Air Transport

Examples of Software Errors


Synchrotron (Particles up to 70 % of light speed)
Ion Source Carbon
Linear Accelerator
Ion Source Proton
Scanning System
Monitor System
Radiation Contro Cross-section through the irradiated tumor volume. Every section represents a different beam range. The treated elements are shown in green
Scanning Magnets
Wire Chambers
Target Volume
Depth 5 cm: oton 80 MeV/u on 145 MeV/u
Depth 10 cm: ton 115 MeV/u on 220 MeV/u

Radio Therapy Machine
software error
** 6 people overdosed**

Year 2010 Bug
30 million debit and credit cards have been rendered unreadable by the software bug

December 4, 2006

The NHTSA said DaimlerChrysler is recalling 128,000 Pacifica sports utility vehicles because of a problem with the software governing the fuel pump and power train control. The defect could cause the engine to stall unexpectedly.

[Washington Post]

software in modern cars
>100K LOC
2006: error in pump control software
** 128000 vehicles recalled**

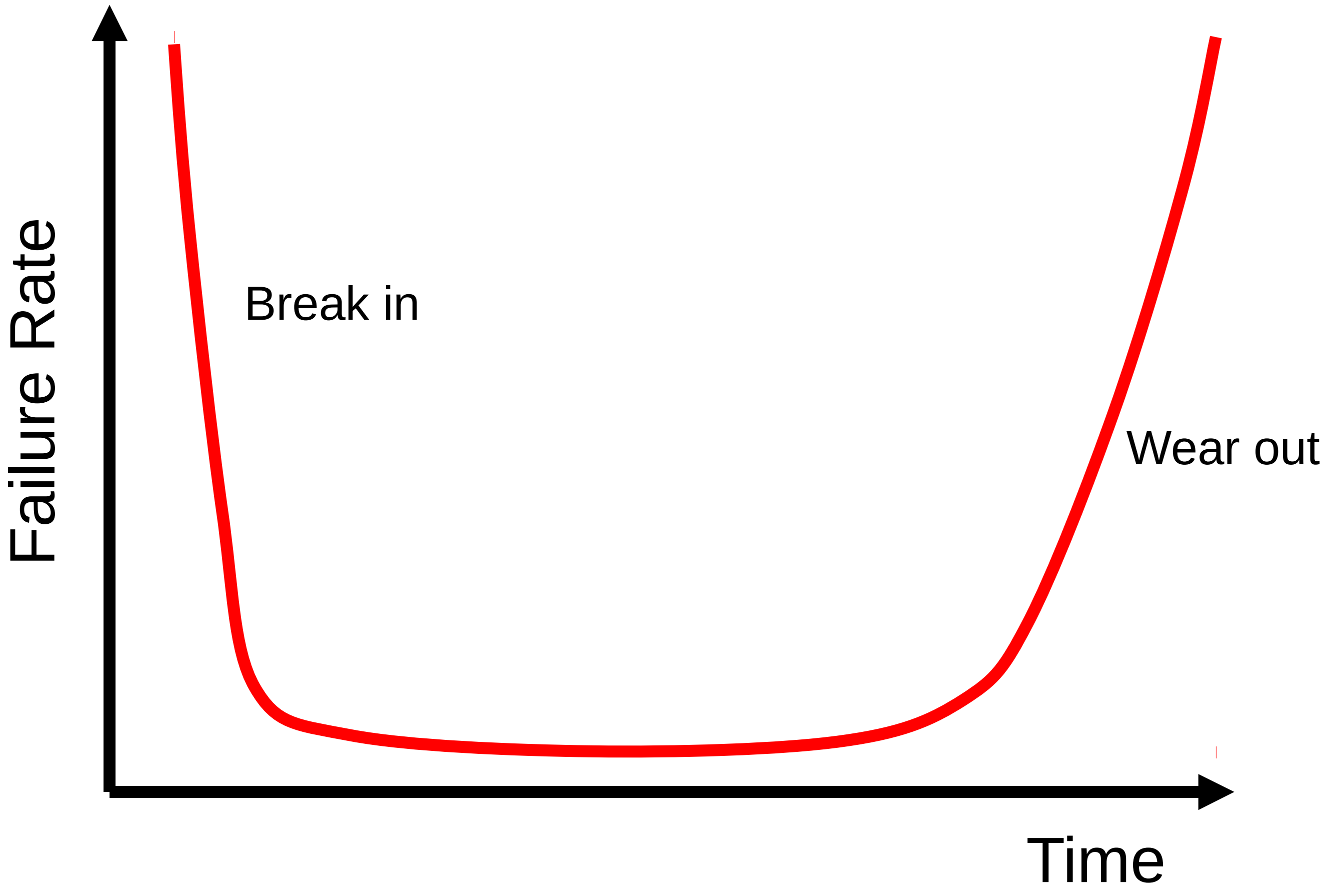
link

Why is Software Development so %$##% Hard? (H)

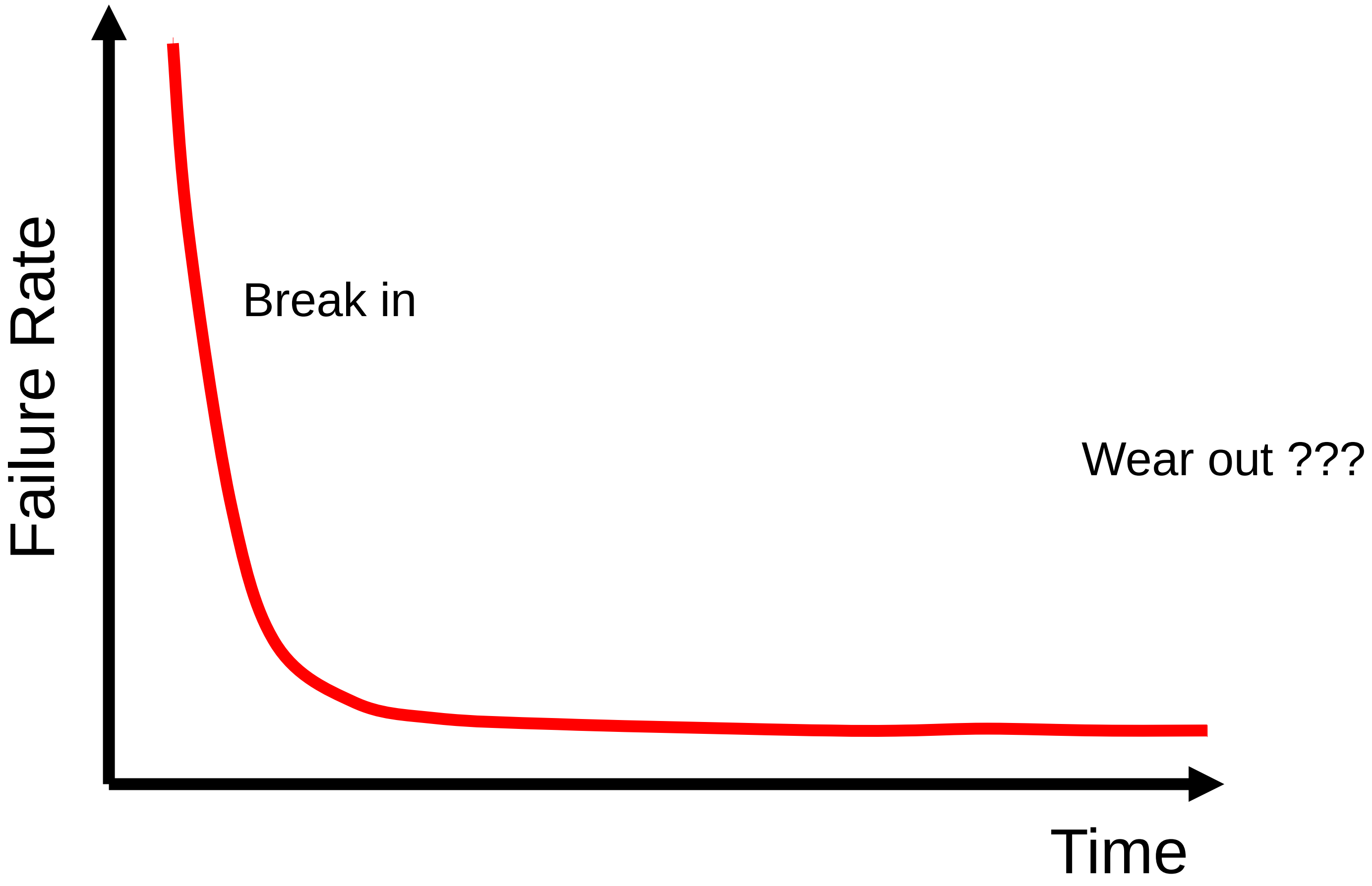
Why is Software Development so %$##% Hard? (L)

- Complexity
  - Software systems are the most complex artifacts ever created
- Invisibility
  - We cannot see the progress of the development
- Changeability
  - Software is “easy” to change
- Conformity
  - The software will have to be molded to fit whatever external constraints may be imposed

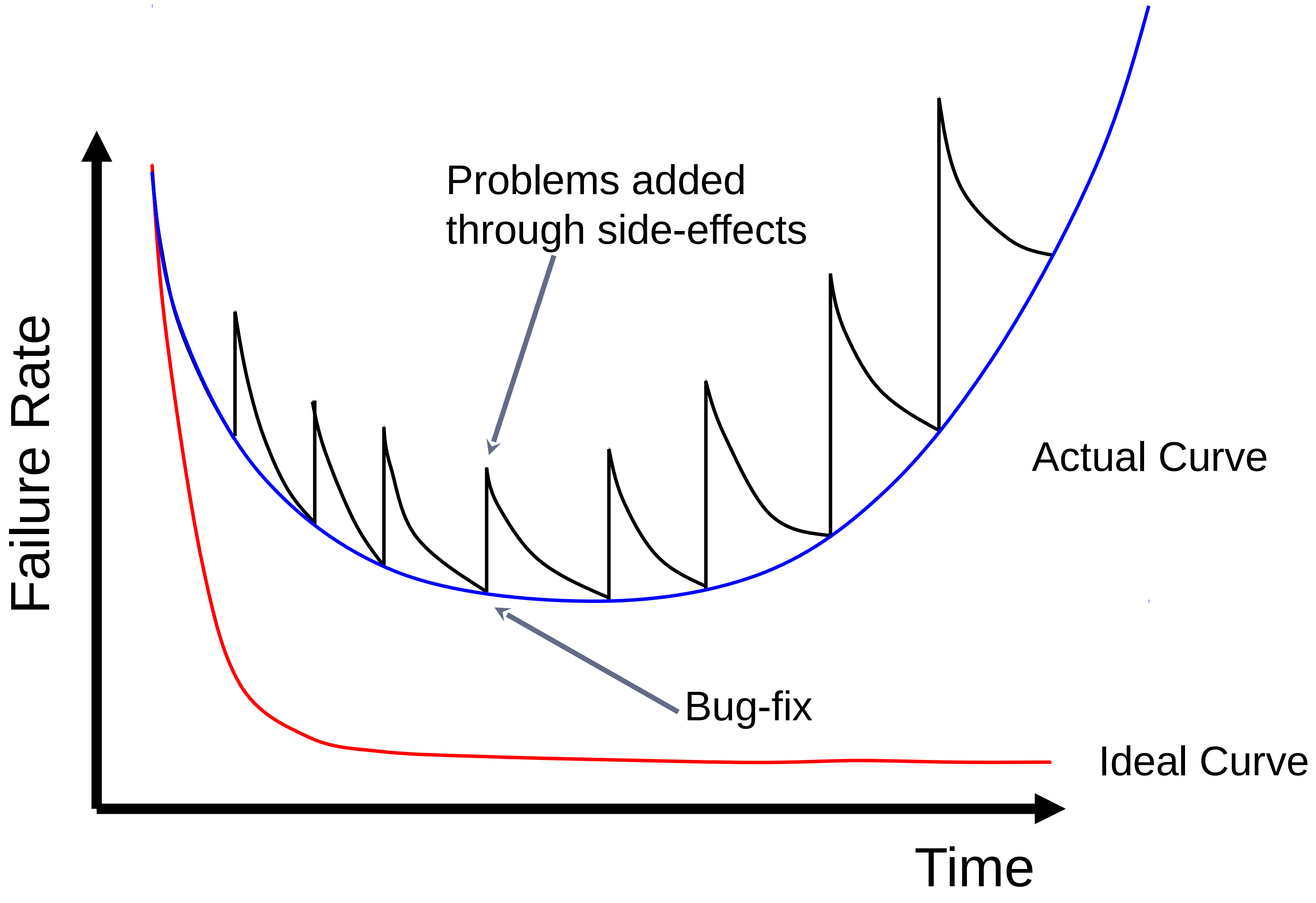
Failure in Hardware



Software Failure



How it Really Works (L)



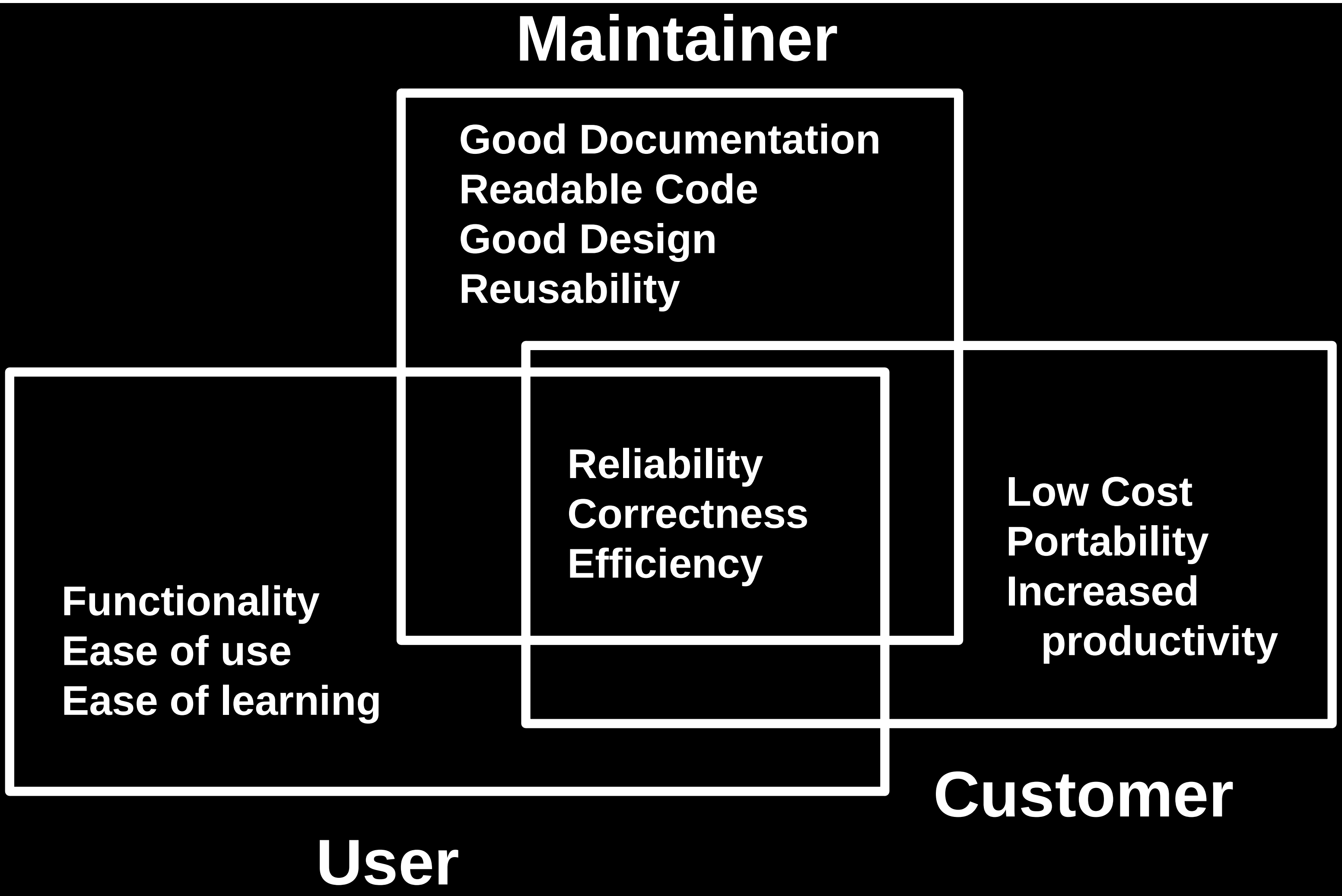
Software As Product

- More than the executable
  - Executable, installation manual, user manual, requirement documentation, design documentation, etc.
- Intangible
- Human-intensive creation
  - Trivial manufacturing process (copying)

Software Product Attributes (sample)

- Maintainability
  - It should be possible for the software to evolve to meet changing requirements
- Dependability
  - The software should not cause physical or economic damage in the event of failure
- Efficiency
  - The software should not make wasteful use of system resources
- Usability
  - Software should have an appropriate user

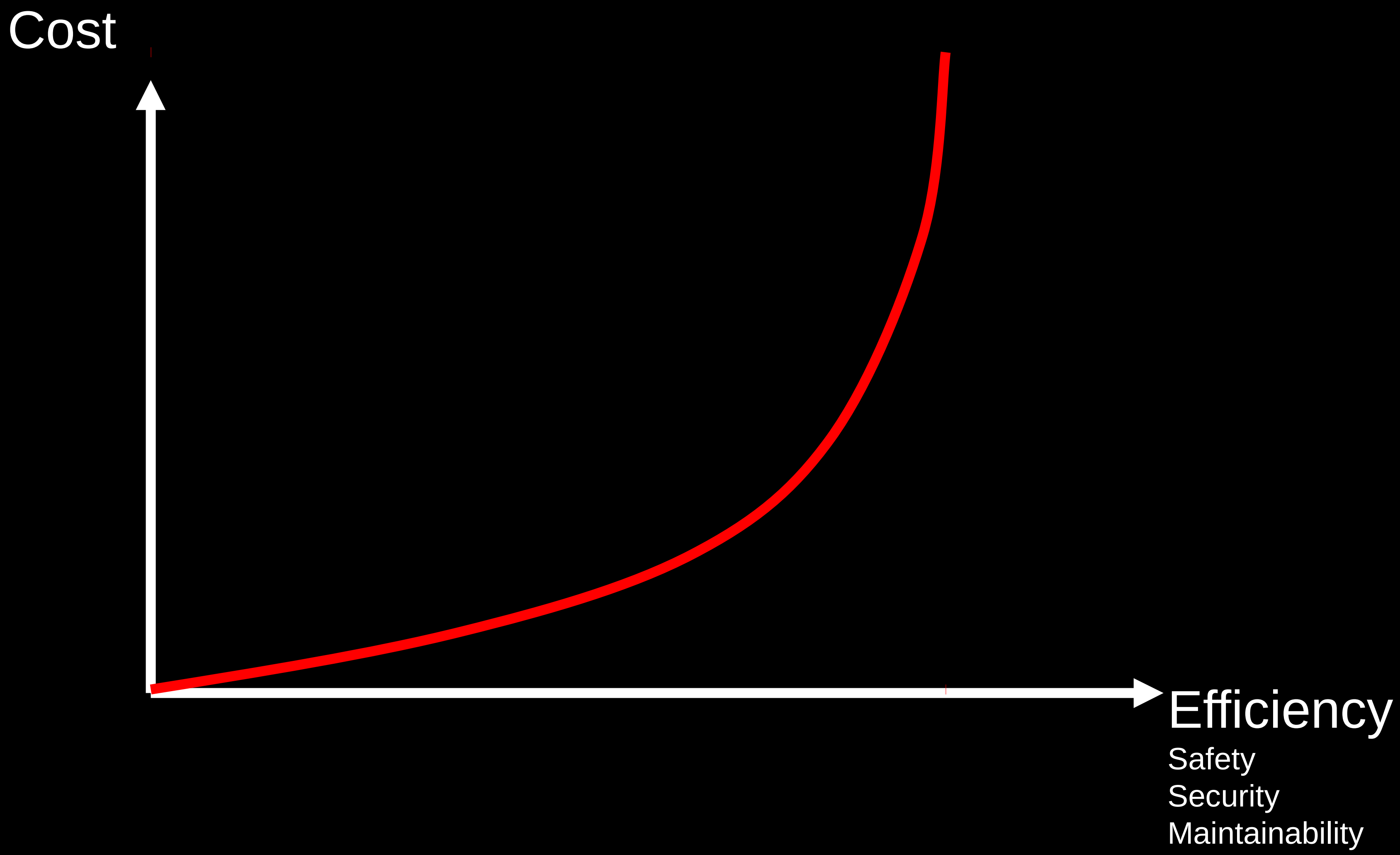
Qualities Are in the Eyes of Beholders



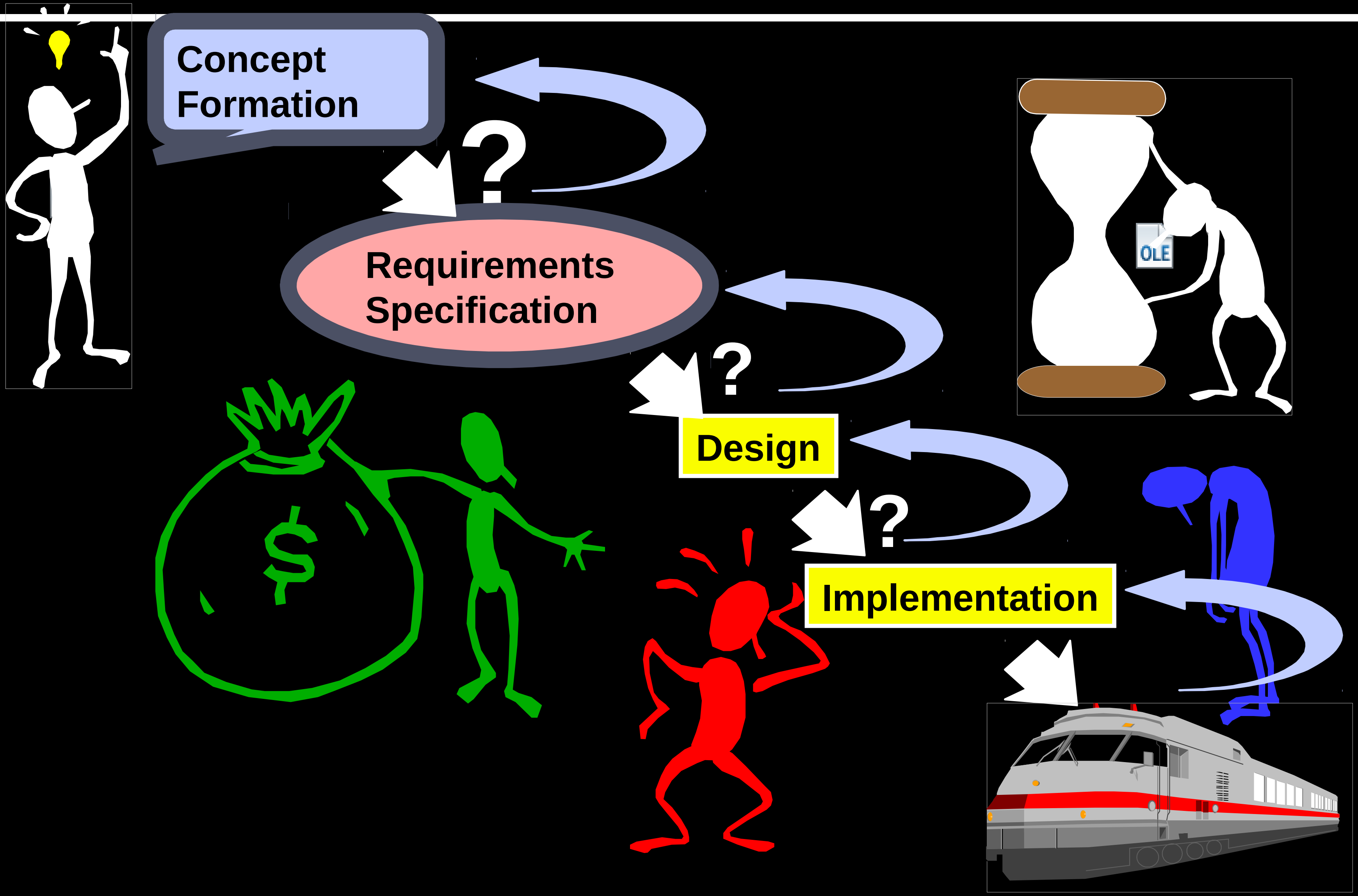
Importance of Product Characteristics

- The relative importance of these characteristics depends on the product and the environment in which it is to be used
- In some cases, some attributes may dominate
  - In safety-critical real-time systems, key attributes may be dependability and efficiency
- Costs tend to rise exponentially if very high levels of any one attribute are required

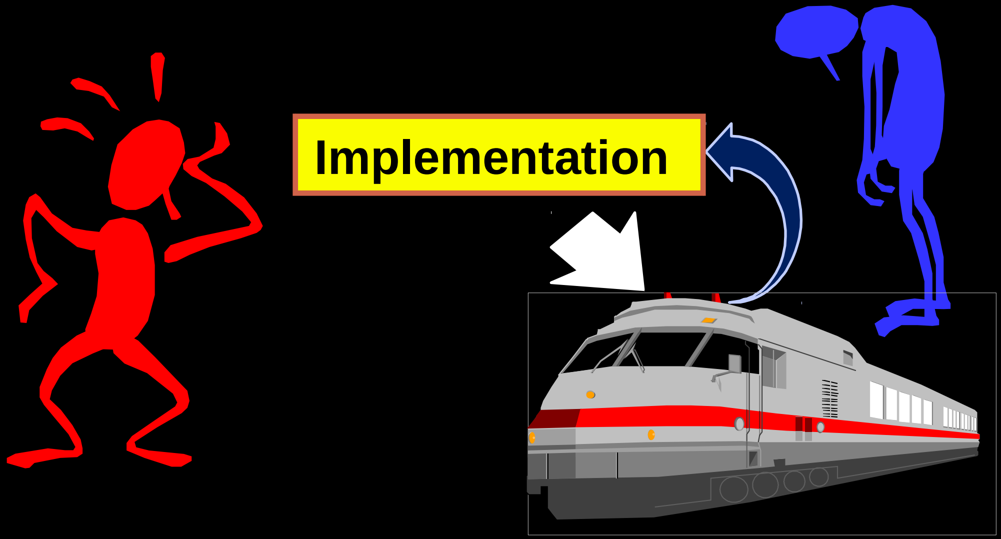
Efficiency Costs



How Software Development Works



Common Process



We Need a Software Process

- Structured set of activities required to develop a software system
  - Specification
  - Design
  - Validation
  - Evolution
- Activities vary depending on the organization and the type of system being developed
- Must be explicitly modelled if it is to be managed

Ethics

- We are very dependent on software in today's world, the dangers of unethical—immoral-- behaviour of software engineers have become more apparent.
- The ACM and IEEE have written a Software Engineering Code of Ethics and Professional Practice:
  - Http://www.acm.org/about/se-code

Ethics

- It all seems simple—until you spot the conflicts. Eg: *Your company depends on a major contract from Client X. Client X insists you use Software Y to develop a product (3.08) on which people's lives depend. You are not satisfied with Y's correctness, and think using it might introduce a risk of life-threatenng failure of the product (1.03). What do you do?*

Lecture Plan (approximate)

Requirements Specification
- 2 lectures

Design Fundamentals
- 1 lecture

Design
- 3 lectures

Coding and version control
- 2 lectures

Testing and Coverage
- 2 lectures

Reliability and Maintenance
- 2 lectures

Process
- 2 lectures

We Have Learned

- What Inf2c-se is about
- What is expected from you
  - Prerequisites
  - Workload
- What software engineering is
- Some of the problems

Next Time

- Note: **No Lecture on Thursday**. Next lecture will be next Tuesday.
- 1st Tutorial is next week.
- The fundamental principles of software engineering
- Requirements and use cases (will be re-visited in tutorial1).
- Read Sommerville Chapters 1 and 4.