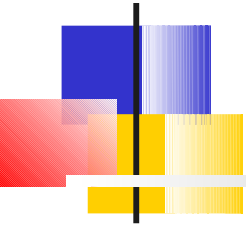


Information Systems Concepts

Problems in Information Systems Development



Roman Kontchakov

Birkbeck, University of London

Based on Chapter 2 of Bennett, McRobb and Farmer:
Object Oriented Systems Analysis and Design Using UML, (4th Edition), McGraw Hill, 2010



Outline

- What Are the Problems?
 - Section 2.2 (pp. 44 – 52)
- Why Things Go Wrong?
 - Section 2.3 (pp. 52 – 56)
- Essence and Accidents of Software Development





An Exaggeration?

70% of software projects fail
(The Standish Group report, 2005)



How the customer explained it



How the Project Leader understood it



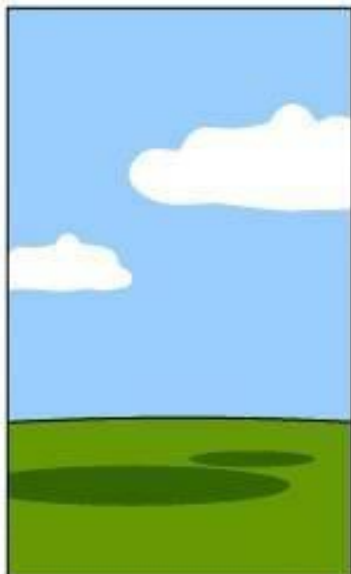
How the Analyst designed it



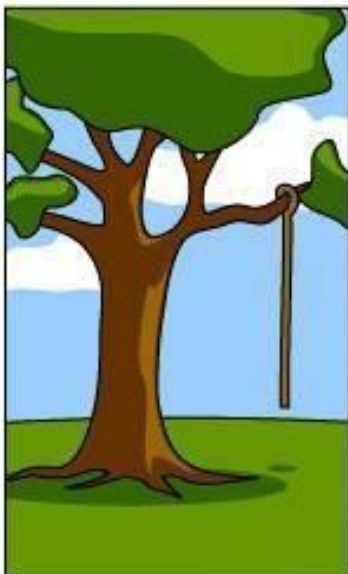
How the Programmer wrote it



How the Business Consultant described it



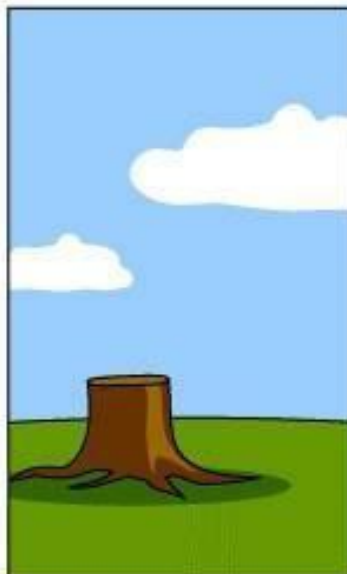
How the project was documented



What operations installed



How the customer was billed



How it was supported



What the customer really needed



Three types of players in IS project

- **end-users**
 - those who will benefit from the system's outputs, directly or indirectly
- **clients**
 - managers, those who have control or influence over the initiation, direction or progress of the project
- **developers**
 - those who are responsible for the development of the IS



End-user's perspective

- “What system? I haven’t seen a system.”
vapourware -- projects that are never finished
- “It might work, but its dreadful to use!”
poor interface, incomprehensible error messages,
unhelpful 'help', poor response time, unreliability
- “It’s very pretty, but does it do anything useful?”



Client's perspective

- "If I'd known the real price, I'd never have agreed."
- "It's no use delivering it now, we needed it last April!"
- "OK, so it works, but the installation was such a mess my staff will never trust it."
- "I didn't want it in the first place."
- "Everything has changed now, we need a completely different system."



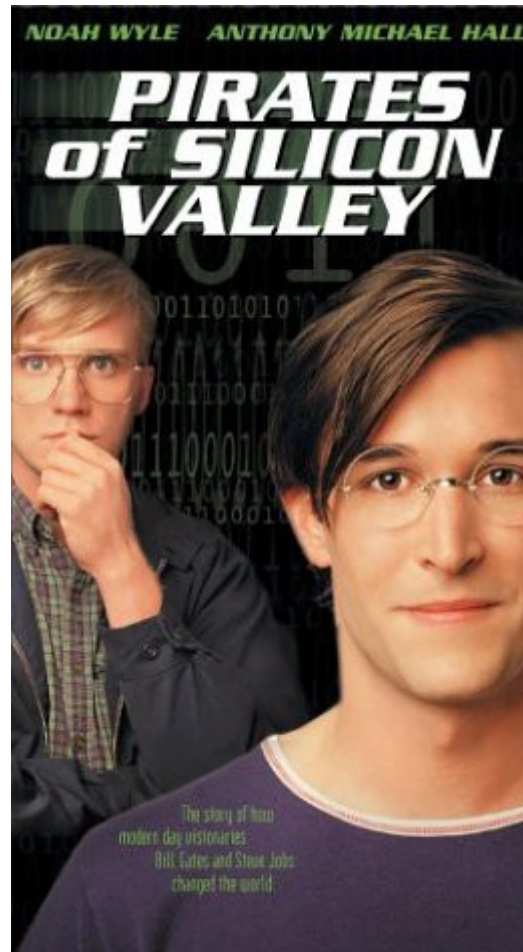
Developer's perspective

- "We built what they *said* they wanted."
- "There wasn't enough time to do it any better."
- "Don't blame me, I've never done OO analysis before!"
- "How can I fix it? I don't know how it's supposed to work."
- "We said it was impossible, but no-one listened."
- "The system is fine. The users are the problem."



Why Things Go Wrong?

We are faster!



We are better!



Why Things Go Wrong?

- Quality Problems
 - The wrong problem is addressed
 - System conflicts with business strategy
 - The context is neglected
 - Organization culture may be ignored
 - The system analysis or design is performed incorrectly
 - Team is poorly skilled or inadequately resourced (e.g., not enough time allowed)
 - The project is carried out for the wrong reason
 - Technology pull or political push (e.g., the dot.com crashes)



Why Things Go Wrong?

- Productivity Problems
 - Customers change their minds about the requirements
 - Requirements drift
 - External events change the environment
 - e.g., new legislation, introduction of the Euro currency, etc.
 - Implementation is not feasible
 - May not be known until the project has started
 - Poor project management
 - Inexperienced management or political difficulties



Geoffrey James: The Tao of Programming (chapter 5.2)

A manager asked a programmer how long it would take him to finish the program on which he was working.

“It will be finished tomorrow,” the programmer promptly replied.

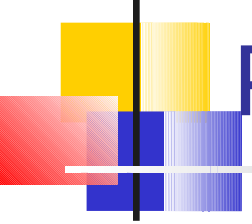
“I think you are being unrealistic,” said the manager, “Truthfully, how long will it take?”

The programmer thought for a moment. “I have some features that I wish to add. This will take at least two weeks,” he finally said.

“Even that is too much to expect,” insisted the manager, “I will be satisfied if you simply tell me when the program is complete.”

The programmer agreed to this.

Several years later, the manager retired. On the way to his retirement luncheon, he discovered the programmer asleep at his terminal. He had been programming all night.



Geoffrey James: The Tao of Programming (chapter 3.4)

A manager went to the master programmer and showed him the requirements document for a new application.

The manager asked the master: “How long will it take to design this system if I assign five programmers to it?”

“It will take one year”, said the master promptly.

“But we need this system immediately or even sooner! How long will it take if I assign ten programmers to it?”

The master programmer frowned. “In that case, it will take two years.”

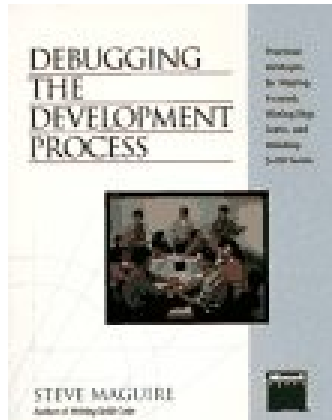
“And what if I assign a hundred programmers to it?”

The master programmer shrugged. “Then the design will never be completed,” he said.



Why Things Go Wrong?

- That Sinking Feeling
 - Debugging the Development Process by Steve Maguire, Chapter 8



If your project is slipping, something is wrong. Don't ignore the causes and demand long hours of the team members. Find and fix the problems.



Brooks' Law

Adding manpower to
a late software project
makes it later.



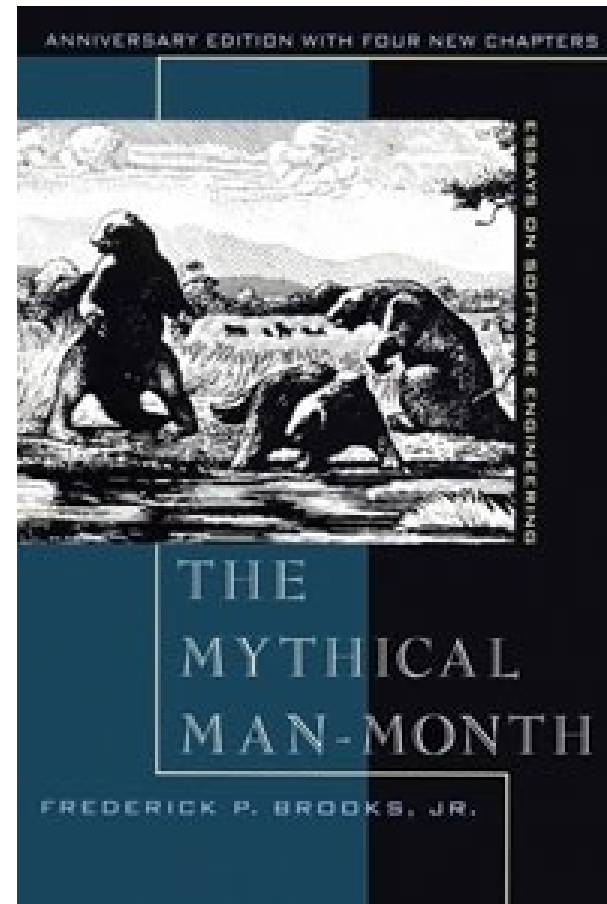
The Mythical Man-Month

- 100 Man-Month?
 - 1 Man x 100 Month
 - 10 Man x 10 Month
 - 100 Man x 1 Month

Group Intercommunication Formula:

$$n(n-1)/2$$

e.g., 50 developers give $50 \cdot (50-1)/2 = 1225$ channels of communication





No Silver Bullet

“Essence and Accidents of
Software Engineering”
by F.P. Brooks





The **essence** of software development

- Difficulties defined by the issues inherent in the software itself
 - software is a product of a creative act (not a result of a repetitive act of manufacturing)
- Software development inherent properties
(*not* amenable to 'silver bullets' or breakthroughs):
 - complexity (no repetitive elements)
 - conformity (to old interfaces, no redesign)
 - changeability (is often used beyond the original domain)
 - invisibility (no geometric representation)



The **accidents** of software development

- Difficulties due to software production practices
- Software development variables: amenable to human intervention
 - attributed mostly to the fact that an information system is a social system
 - the software solution must not be adding to the inherent complexity of the software product
 - adaptiveness (supportability) is the challenge
 - = understandability + maintainability + scalability (extensibility)
 - related to
 - stakeholders, process and modelling



Linus' Law

Given enough eyeballs,
all bugs are shallow.





Lines of Code

Measuring programming progress by lines of code is like measuring aircraft building progress by weight.





Take Home Messages

- What Are the Problems?
 - 3 types of main players → 3 perspectives
- Why Things Go Wrong?
 - Quality Problems
 - Productivity Problems
- Essence and Accidents of Software Development