During Introduction:

•Name, Number of years in Software Development, Role now.

•Problems faced on this project or previous. Make a list. At the end, most problems should be solved.

•How familiar are you with Scrum? 10 == read both books; I know what it's about; I've practiced it! 1 == seemed kinda interesting so I decided to take a training course

•What development model best describes your project working environment? 10 == consulting: firm is working with external clients; client driven development. 1 == corporate: managing development of in-house product or service

Contents

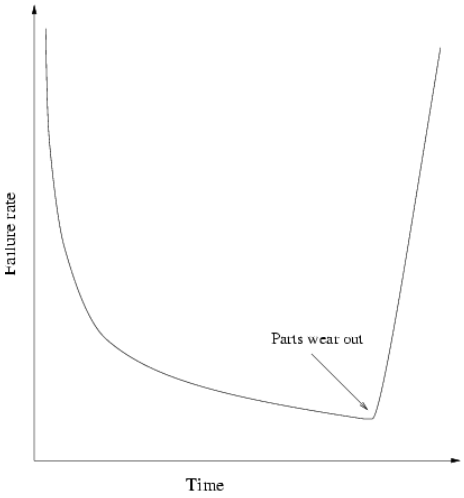
Introduction

Listening, Designing, Coding, Testing –
That's all there is to software. Anyone, who
tells you different, is selling something

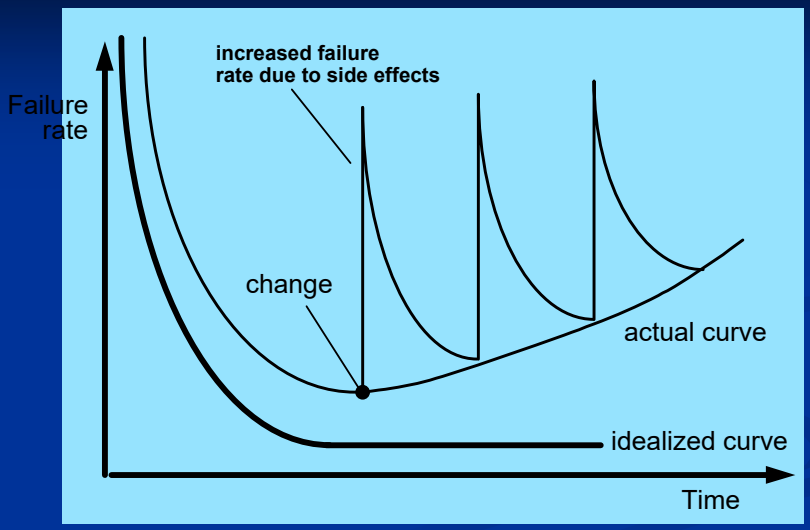
Software is Different

- Software is developed not manufactured
  - The biggest cost is personnel
- Software doesn't wear out
  - 60 – 85% of software costs are in modifications.
- Most software is custom built.
  - Software is not just a combination of spare parts or components.

Hardware wear



Software Deteriorates



The software development is a complex process

Delivering software is hard, and delivering quality software in time is even harder.

For many large clients, the downtime costs more than $100,000 per hour. So 98% uptime is not good enough. Customers want 99.999% uptime

Java SE has 6 million LOC.

> According to Washington D.C., National Institute of Standards and Testing, software errors cost U.S. economy $60 billion per year in 2002. Finding defects earlier can save $22 Billion / year.
>
> This 60 billion is the cost of developers & users. It does not include lost revenue or productivity.

In the past, good software means fast and small.

Now, good software means that which can be changed easily.

> For maintenance of software, developers spend most time in integrating new features and not in coding new features.

Example of Failure

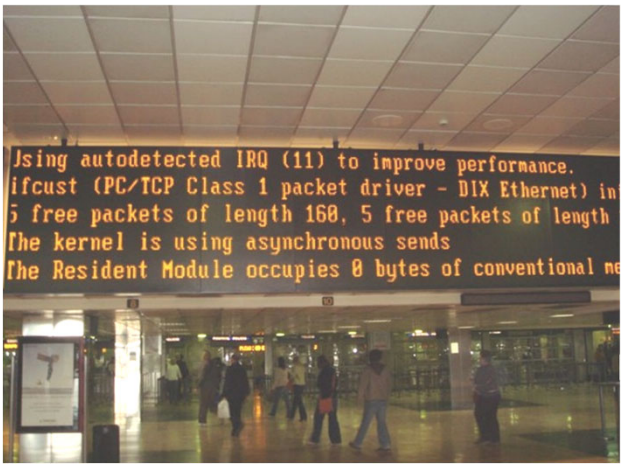
Dell's ERP system.

Dell spent $200 million and then decided their ERP system wasn't going to be flexible enough for them and canned it and started again.

Luxury cars of Today

- Today's typical cars contain
  - 30-100 microprocessors
  - An average of 35 million lines of code
- Computers cause a third of all warranty claims
- 90% of future vehicle innovations will come from computers

Madrid Airport



When failures occur, a common tendency is for processes to become either more chaotic or more bureaucratic.

•Both approaches fail to meet intended objectives, as they cannot respond to change in timely, economic, transparent and measurable manner.

•Feedback and opportunity are often ignored. People are prone to repeating the mistakes of the past, often with unjustified sense of optimism – “Things will be different this time”.

Typical IT Budget

This is the problem

New Projects
<20%

Maintenance and Operations
>80%

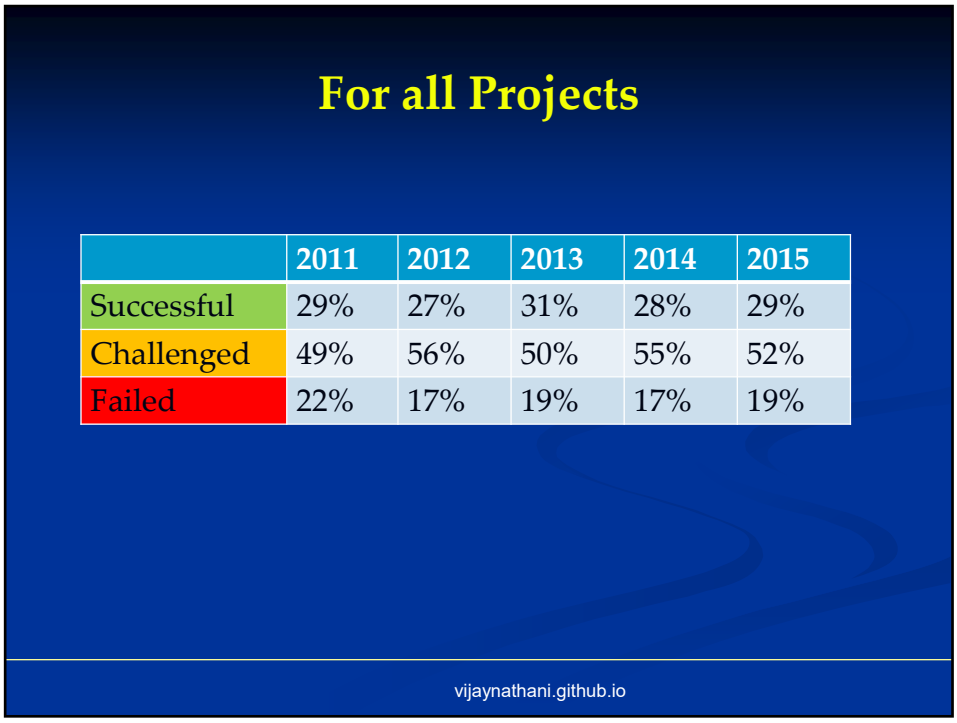
Most organizations don't scrutinize M&O expenditures
No other major expenditure has so little oversight

Source Burton group. Presentation on Infoq.com on SOA

Problems in Software Development

- Why does it take so long?
- Why is cost so high?
- Why can't we find all errors before we give the software to our customers?
- Why do we spend so much time and effort maintain existing programs?
- Why do we continue to have difficulty in measuring the progress as software is being developed and maintained?

Businesses are finding too often that their software systems act as brakes on their competitiveness, rather than as accelerators. Businesses are discovering that while their markets change rapidly, their software systems do not.



Reference: http://www.infoq.com/articles/standish-chaos-2015

Explain that these stats are from the Standish Group (formed in the mid 1980s). This group started with a specific remit to gather statistics and analyze data on real-life s/w engineering projects to identify specific factors that relate to success and failure.
In 1994 the Standish Group published the Chaos Report – one of the most often quoted sources about the overall track record of s/w development projects. The report was based on over 8000 projects across a variety of different industry sectors. The report suggested that a staggering 31% of all s/w development projects were failures. Only 16% of projects were completed on time and on budget with all features envisioned.

In 2004 the Standish Group published an updated report based on the analysis of an additional 40,000 projects in the previous 10 years. The good news is that project success has more than doubled from 16% to 34% and project failure has dropped by more than half, from 31% to 15%.

Although the trend is moving in the right direction there is still an awful long way to go. A 34% success rate still indicates that 66% of software projects still fail! Very few other industries would accept that level of performance.
==========

Standish group statistics.

Successful: “Completed on time, on budget, with all features and functions as

originally specified."

Challenged: "Completed and operational but over budget, over the time estimate, [with] fewer features and functions than originally specified."

Failed: "Cancelled at some point during the development cycle."

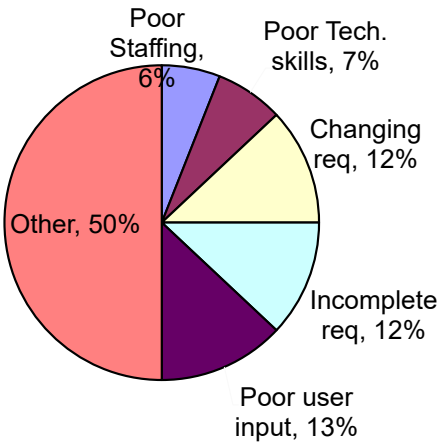
Despite their popularity, there's something wrong with these definitions. A project can be successful even if it never makes a dime. It can be challenged even if it delivers millions of dollars in revenue. *CIO Magazine commented on this oddity:*

Projects that were found to meet all of the traditional criteria for success—time, budget and specifications—may still be failures in the end because they fail to appeal to the intended users or because they ultimately fail to add much value to the business.

... Similarly, projects considered failures according to traditional IT metrics may wind up being successes because despite cost, time or specification problems, the system is loved by its target audience or provides unexpected value. For example, at a financial services company, a new system... was six months late and cost more than twice the original estimate (final cost was $5.7 million). But the project ultimately created a more adaptive organization (after 13

months) and was judged to be a great success—the company had a $33 million reduction in write-off accounts, and the reduced time-to-value and increased capacity resulted in a 50 percent increase in the number of concurrent collection strategy tests in production.

Reasons for Problems



Impact

- The crisis persists
- It is not a crisis but a chronic problem

After 40 years later, the software "crisis" is still with us

Major problems are still the same:

poor quality (correctness, usability, maintainability, etc)

over budget

delivered late, or not at all

==========

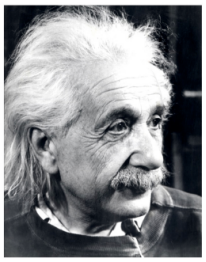
It becomes a persistent, chronic condition that software industry has to face with

A study done in 2005, mentions that number one complaint of Java software development managers is their inability to minimize bugs in code before it is released.

Myths

- Management
- Customer
- Developer

"NO PROBLEM CAN BE SOLVED FROM THE SAME LEVEL OF CONSCIOUSNESS THAT CREATED IT."



Management Myths:

•Value is a forward looking proposition. It cannot be measured in rear view.

•We have a book that is full of standards and procedures for building software.

• My people know everything they need to know.

•If we get behind schedule, we can add more programmers and catch up.

•We have outsourced the project. We can relax and let the firm build it.

•Our code base must be worth a lot, because of the cost involved in producing it

Customer Myths

•A general statement of objectives is sufficient to begin writing program. We can fill in the details later.

•Project requirements continually change, but change can be easily accommodated because software is flexible.

Developer Myths

•Meeting customer needs is easy, etc

•Once we write a program and get it work, our job is done.

•Until I get the program running, I have no way of assessing its quality.

•The only deliverable work product for a successful project is the working program.

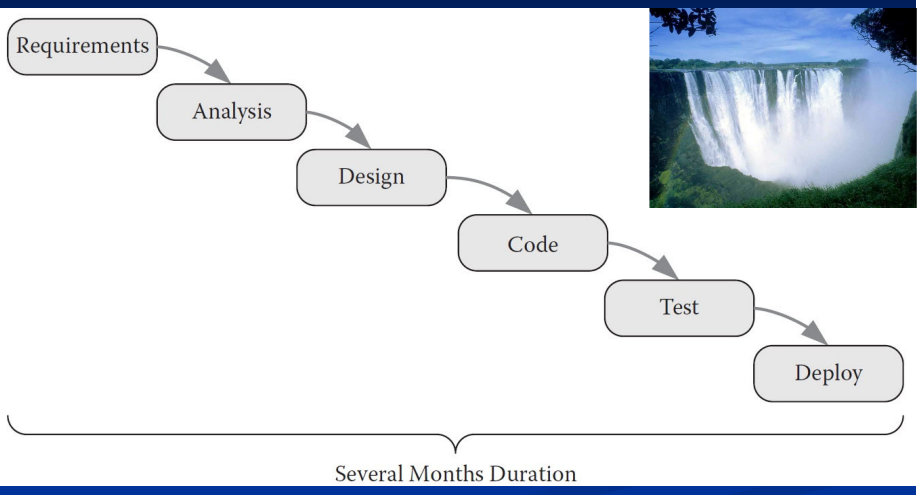
•Software process will make us create voluminous and unnecessary documentation and will invariably slow us down.

Einstien – Change level of consciousness to solve the current problem.

Predictive Processes

He, who chooses the beginning of a road,
chooses the place it leads to.

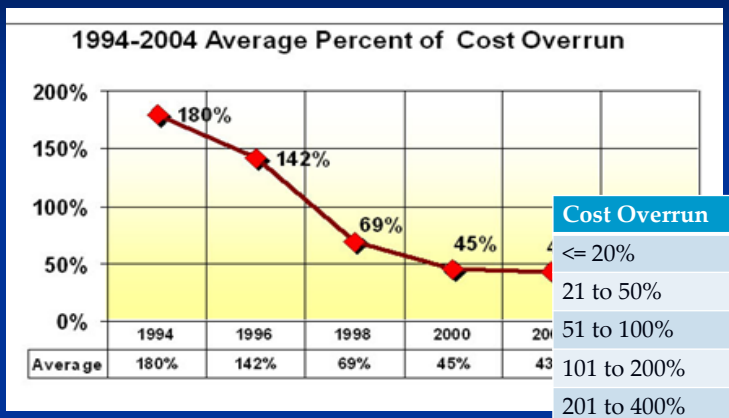
The Waterfall Model



Effort Allocation (Sample Data)

- "front end" activities
  - customer communication
  - analysis
  - design
  - review and modification
- construction activities
  - coding or code generation
- testing and installation
  - unit, integration
  - white-box, black box
  - regression

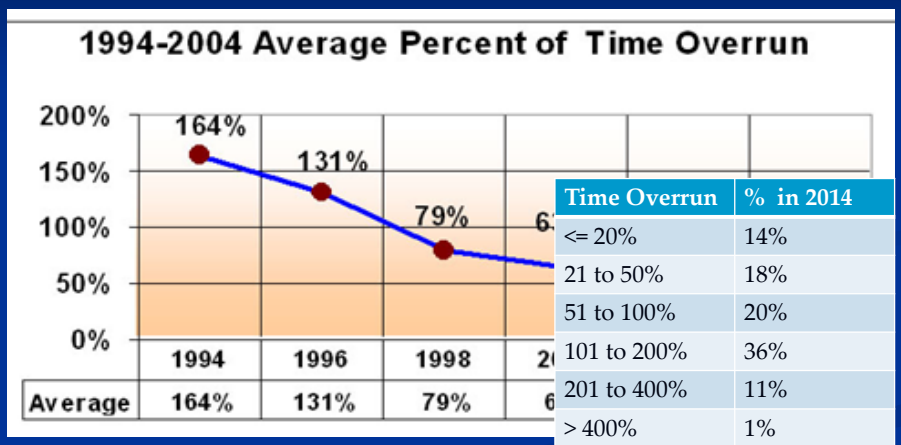
Cost Overruns over Original Estimate



Cost Overrun	% in 2014
<= 20%	16%
21 to 50%	31%
51 to 100%	30%
101 to 200%	10%
201 to 400%	9%
> 400%	4%

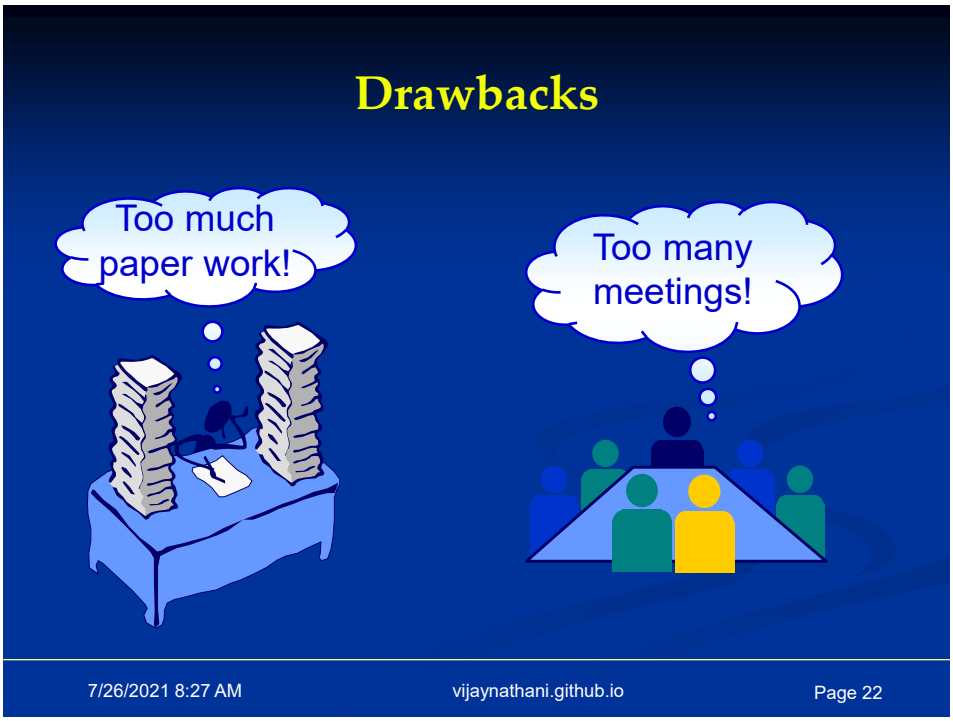
Cost overruns table as per 2014 CHAOS report from Standish group

Time Overruns over Original Estimate



Time Overrun	% in 2014
<= 20%	14%
21 to 50%	18%
51 to 100%	20%
101 to 200%	36%
201 to 400%	11%
> 400%	1%

Time overruns table as per 2014 CHAOS report from Standish group



Studies of documentation produced during conventional software development indicated that for an average 1,000-function point software project (about 125,000 lines of code):

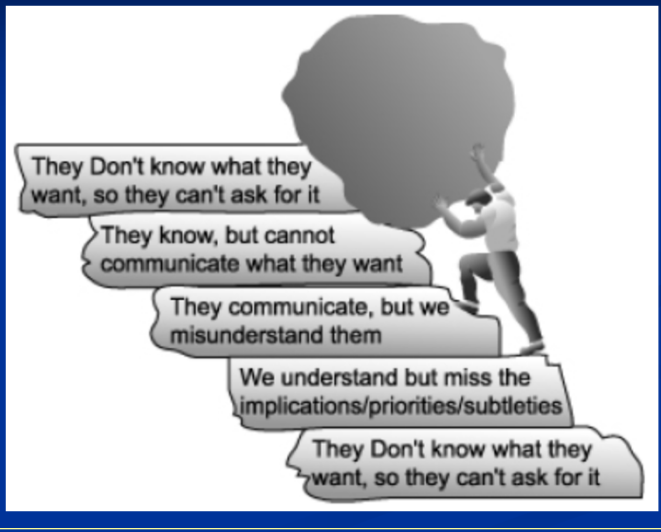
- Requirements documents average 300 pages
- Plans average 100 pages
- Design documents average more than 1,500 pages
- User manuals average more than 600 pages
- Test reports average more than 5,000 pages

The mere presence of a detailed specification may act to the detriment of cooperation between the parties, encouraging both parties to hide behind the specification rather than seeking mutual beneficial solutions

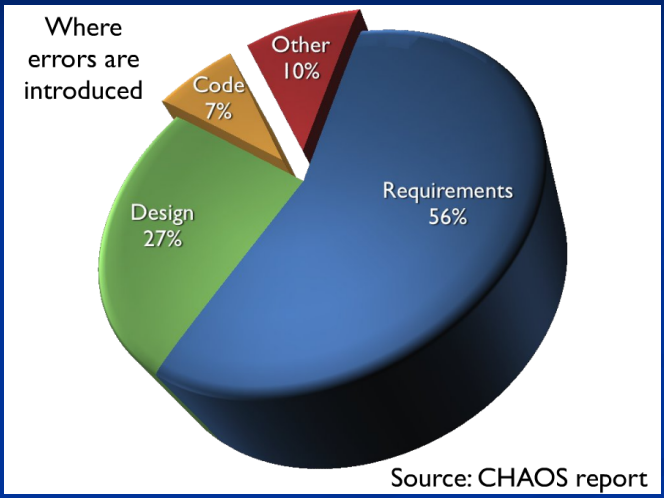
Example of paperwork:

Excessive documentation does not add value but only eats up resources and time.

Unclear Requirements

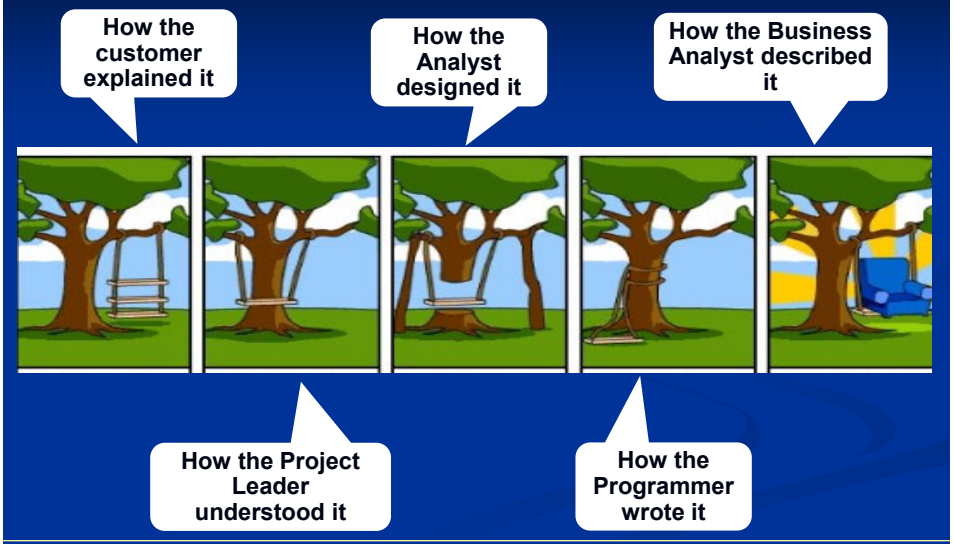


Source of Errors

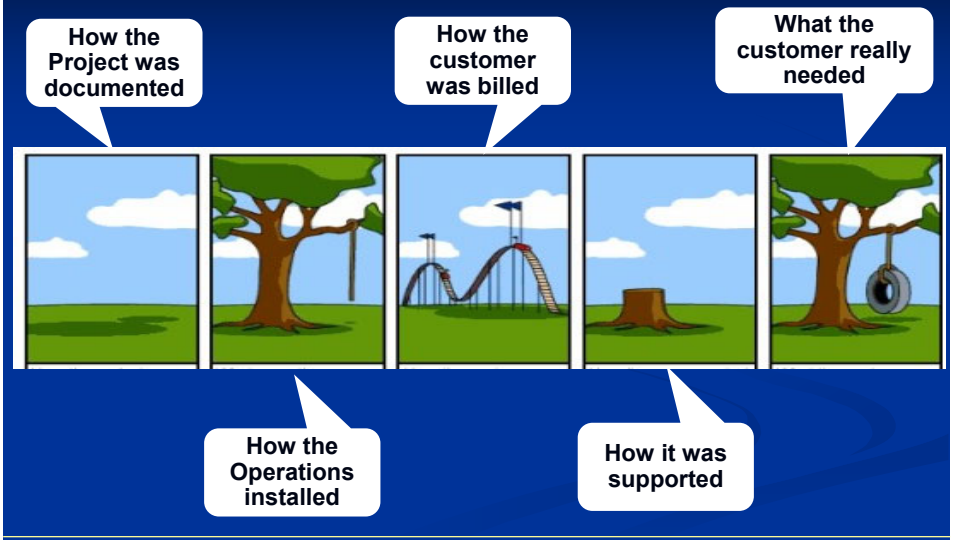


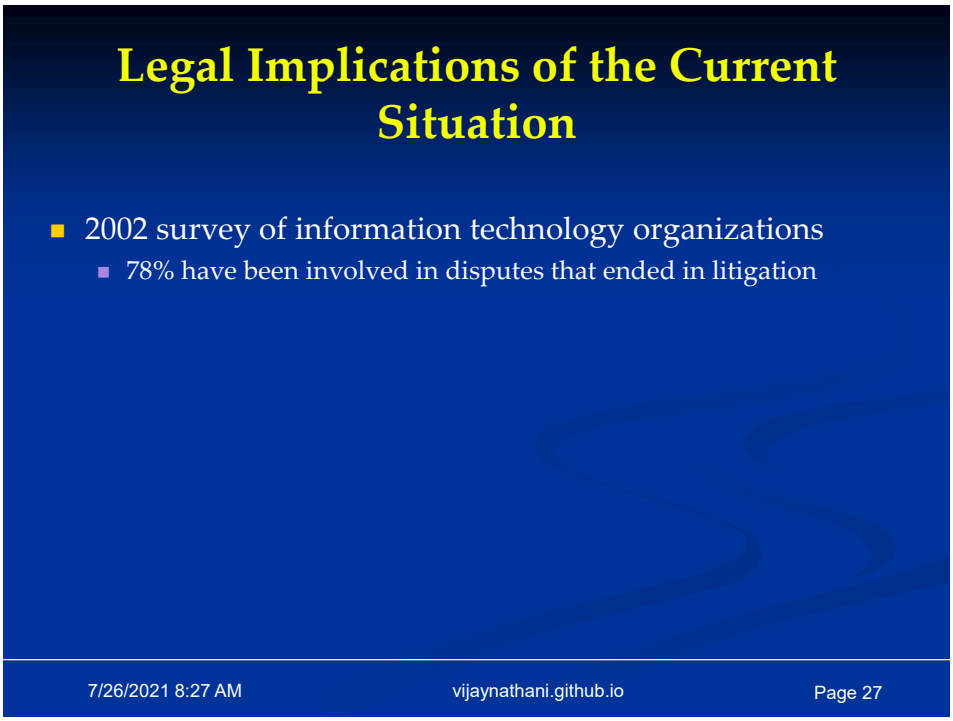
See: http://www.infoq.com/presentations/hellesoy-bdd-rspec

Software Construction …



Software Construction …





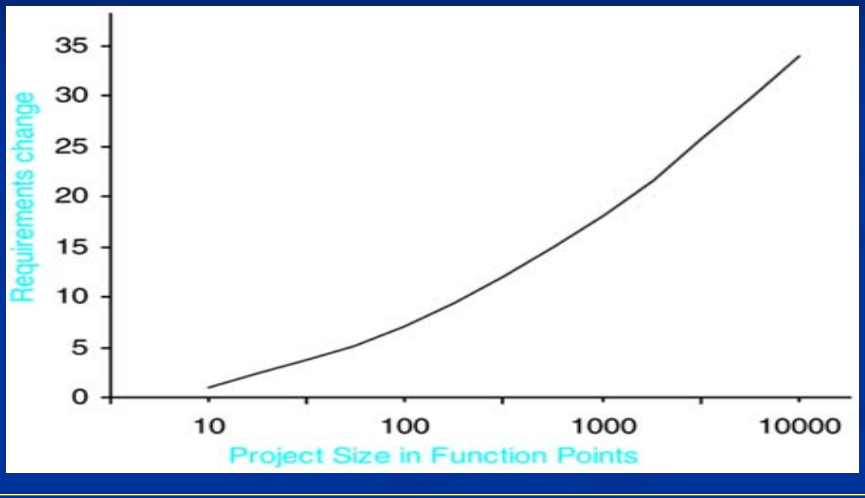
For the organizations that entered into litigation:

In 67% of the disputes, the functionality of the information system as delivered did not meet not up to the claims of the developers

In 56% of the disputes, the promised delivery date slipped several times

In 45% of the disputes, the defects were so severe that the information system was unusable

How to handle change?



Software development is about change:

•Changing requirements

•Change in technology

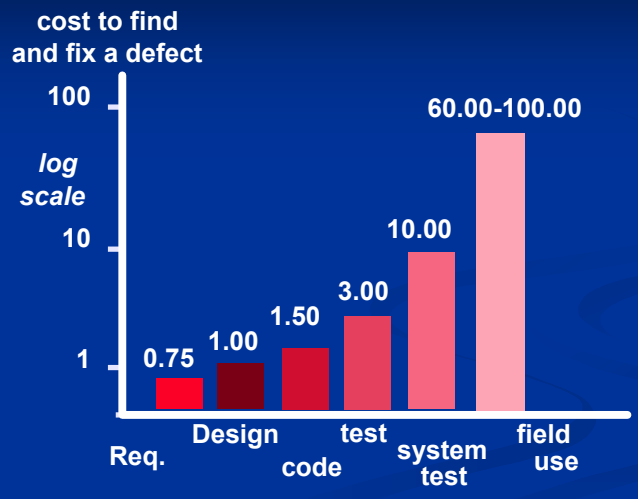
•Change in people involved

•Change in competition, customer, government rules

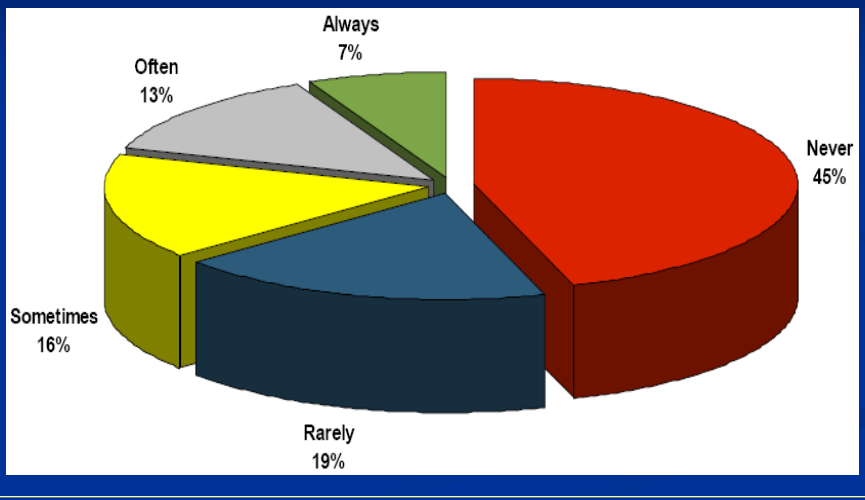
Planned development is like planned economy.

As per 2014 CHAOS report from Standish group. On Small projects, 24% features are due to change requests in end product; 35% for medium projects and 39% for large projects.

Cost of Change (COC) rises Exponentially



YAGNI violated



Standish group study reported at XP2002 by Jim Johnson.

Truth

- Customers are afraid
- Developers are afraid
- Management is afraid

Customers are afraid

•They won't get what they asked for.

•They'll ask for the wrong thing.

•They'll pay too much for too little.

•They must surrender control of their career to techies, who don't care.

•They won't ever see a meaningful plan.

•The plans they do see will be fairy-tales.

•They won't know what's going on.

•They'll be held to their first decisions and won't be able to react to changes in the business.

•No one will tell them the truth.

Developers are afraid

•They will be told to do more than they know how to do.

•They will be told to do things that don't make sense.

•They will be given responsibility without authority.

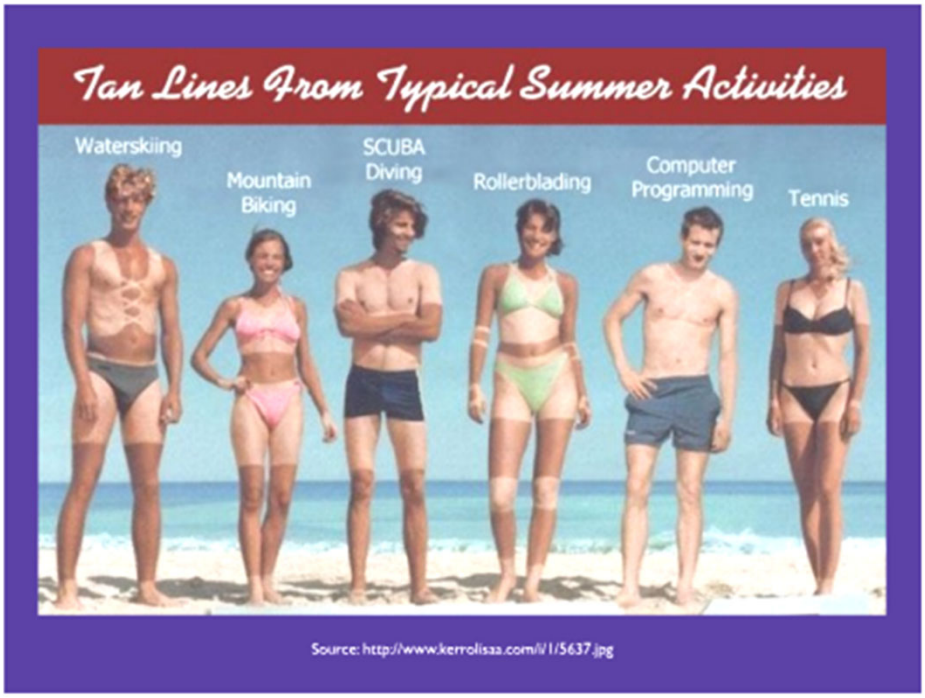
•They won't be given clear definitions of what needs to be done.

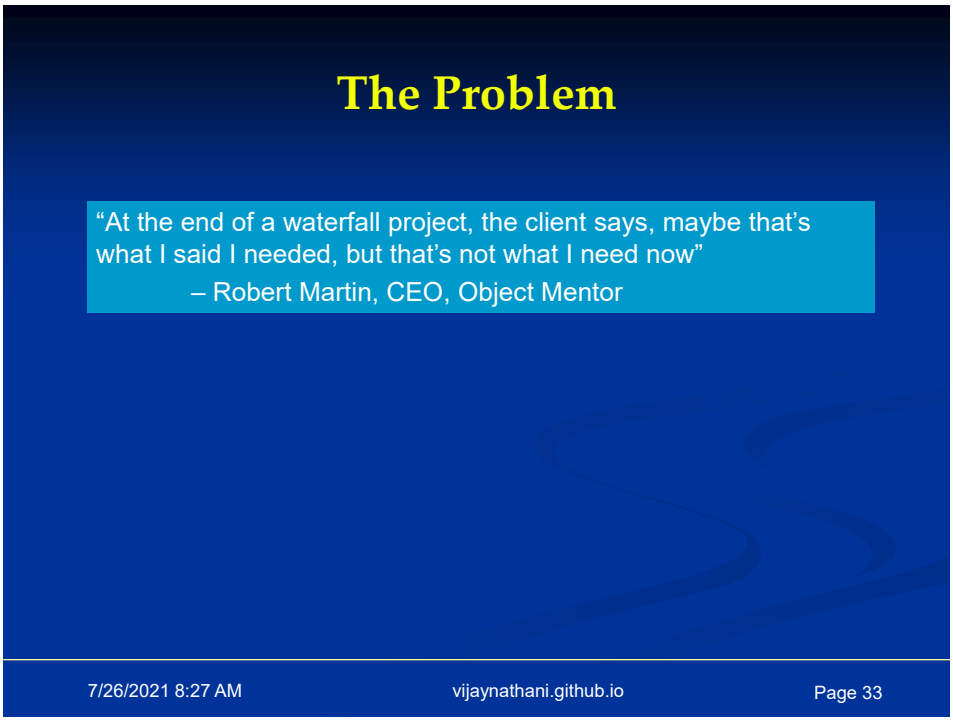
•That they'll have to sacrifice quality for deadlines.

•That they'll have to solve hard problems without help.

•That they won't have enough time to succeed.

- The project will produce the wrong product


Tan Lines From Typical Summer Activities
Waterskiing
Mountain Biking
SCUBA Diving
Rollerblading
Computer Programming
Tennis
Source: http://www.kerrolissa.com/i/1/5637.jpg



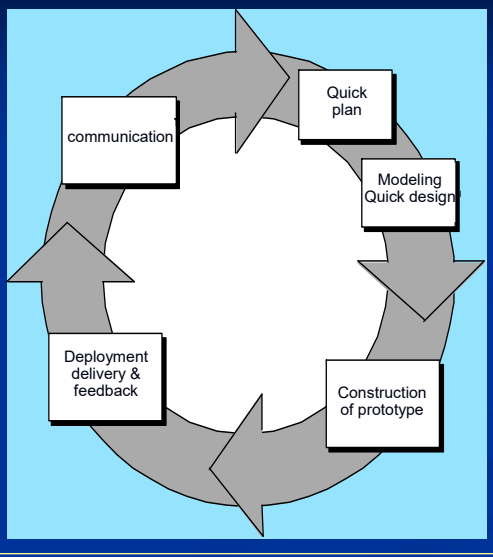
Both the developer and the customer are battered and bloody, if not dead, after the typical project.

> And the project delivers sub-par software at best. Usually it’s junk that everyone is at least a little bit ashamed of, in private.

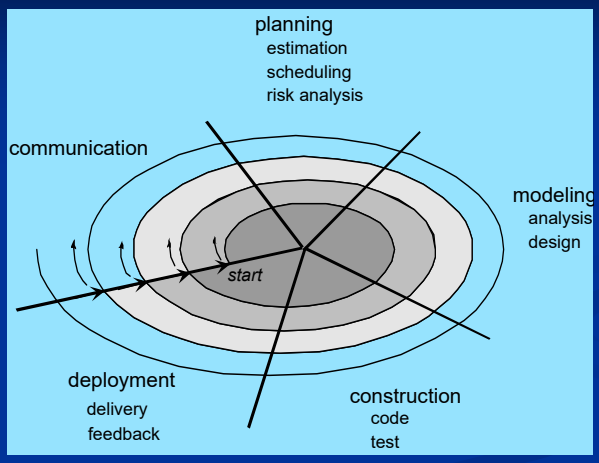
The software industry is in a sorry state, at least from the standpoint of the people who are part of it.

> Developers hate life.
>
> Customers rarely like what they get.
>
> The software stinks.

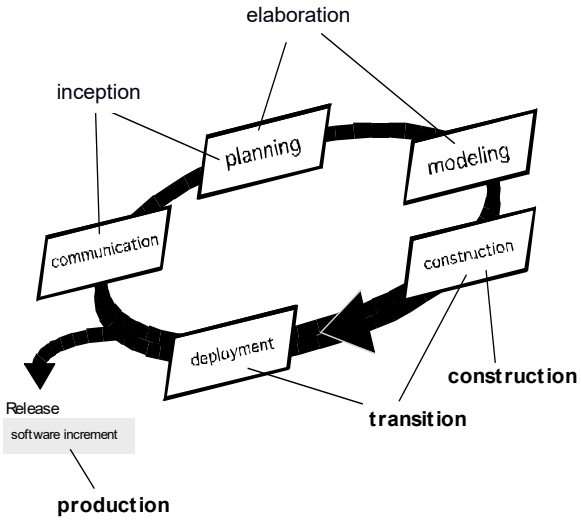
Evolutionary Models: Prototyping



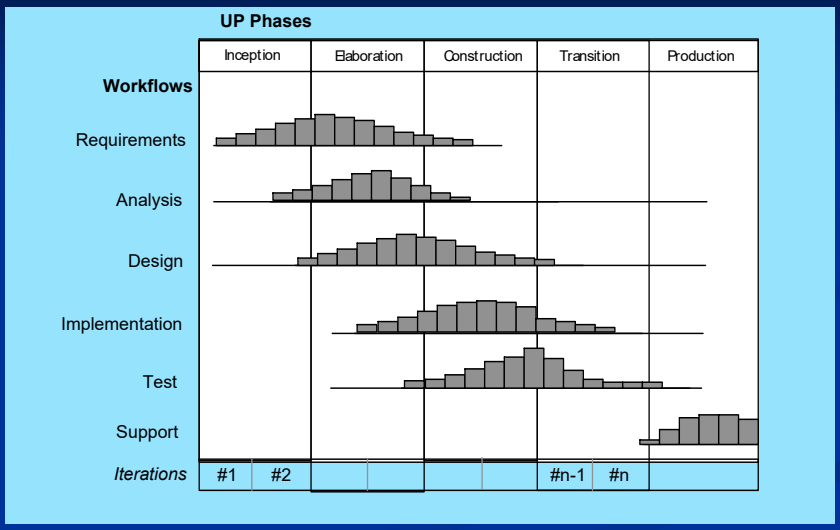
Evolutionary Models: The Spiral



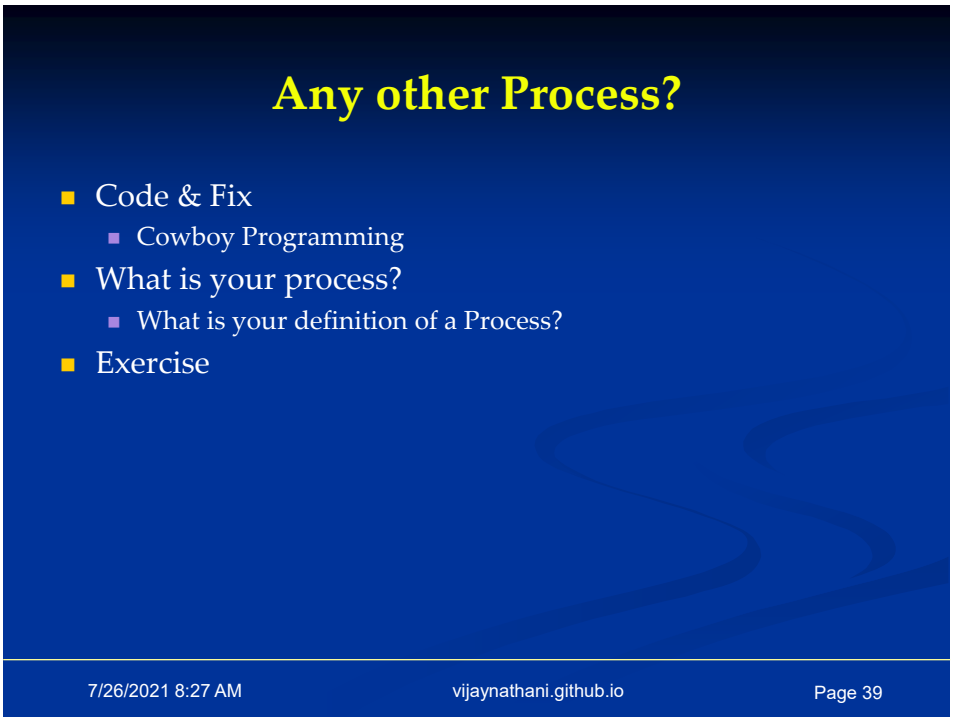
The Unified Process (UP)



UP Phases


UP Phases
Inception
Elaboration
Construction
Transition
Production
Workflows
Requirements
Analysis
Design
Implementation
Test
Support
Iterations
#1
#2
#n-1
#n

Unified Process



Definition of Process: What you do under pressure is your process, everything else is window dressing or political correctness.

All methods are someone else's idea about what you should do when you develop software. It may be useful, from time to time, to borrow from those ideas and integrate them into your own style. It is essential, however, to transcend any method, even your own idiosyncratic method, and "just do it."

Software development is like riding a surfboard—there is no process that will assure a successful ride, nor is there any process that will assure that you will interact propitiously with the other surfers sharing the same wave. Published processes, like published methods, provide observational data from which you can learn and thereby improve your innate abilities—just as observation of master surfers enables you to improve yourself.

==========

Methodology is a social construction. Your "methodology" is everything you regularly do to get your software out. It includes who you hire, what you hire them for, how they work together, what they produce, and how they share. It is the combined job descriptions, procedures, and conventions of everyone on your team. It is the product of your particular ecosystem and is therefore a unique construction of your organization.

All organizations have a methodology: It is simply how they do business. Even the proverbial trio in a garage has a way of working away of trading information, of separating work, of putting it back together all founded on assumed values and cultural norms. The way of working includes what people choose to spend their

time on, how they choose to communicate, and how decision-making power is distributed.