
So, you're applying for a position of web-designer.

SOFTWARE ENGINEERING II
COMP319

Sebastian Coope
coopes@liverpool.ac.uk

Dominik Wojtczak
d.wojtczak@liverpool.ac.uk

My Backgroud

Dominik Wojtczak:

- CS & Maths at University of Warsaw
- PhD at University of Edinburgh
- Postdoc at CWI Amsterdam and University of Oxford
- research interests: automata theory, game theory, probabilistic systems and verification
- COMP313: Formal Methods

My Backgroud

Dominik Wojtczak:

- JavaTech (dedictated web applications using Java Spring MVC, Hibernate, AOP)

- HP Research Labs (filesystem for MemorySpot http://news.bbc.co.uk/1/hi/technology/5186650.stm)

Delivery

- **Lectures**
  - **3 Hours/week**
- **Tutorials (weeks 2 – 11)**
  - **1 Hour/week**
  - work in groups of 2 or more, each tutorial exercise sheet (6 of them) should take around 2 weeks.

Assessment

- 100% written examination in January

- Examination based on
  - Material delivered in lectures
  - Work covered in tutorials

Module aims

- Introduce advanced software engineering topics
- Review and analyses research papers in software engineering

Module materials on VITAL

- Lecture notes
- Selected papers
- Past exam papers and model answers
- Tutorial sheets

Recommended books

- No single recommend text book
  - as the subject is essentially based on research and advanced topics
- But try the following
  - Java Design Pattern Essentials
    - ISBN 0956575803
  - Learning Agile: Understanding Scrum, XP, Lean, and Kanban
    - ISBN 978-1449331924
  - Actors: A Model of Concurrent Computation in Distributed Systems
    - ISBN 978-0262511414

Contents

- Software engineering crisis
- Software cost estimation and project management
- OO design patterns
- XP and Agile
- Dependency graphs and program slicing

Lecture schedule provisional

- Week 1
  - Introduction, software crisis
- Week 2
  - Software crisis (cont.), project management, estimation
- Week 3
  - Cost estimation, OO object patterns
- Week 4
  - OO object patterns cont, Actor model
- Week 5
  - AOP

Lecture schedule

- Week 6
  - Agile open development
- Week 7
  - Agile and Scrum
- Week 8
  - Advanced OO in Javascript
- Week 9
  - Quality management and slicing
- Week 10
  - Revision week

Software Engineering

- Highly complex
  - Many platforms
    - iPhone, Android, Blackberry, PC, Linux, HTML5 etc. etc.
  - Many models
    - Standalone, client-server, peer-to-peer
  - Hard problems
    - AI, neural nets
  - Large size
    - Linux kernel >10 million lines

Software crisis

- Catastrophic
  - Ariane 5 crashed due to variable overflow in software
  - cost 7 billion USD
  - https://www.youtube.com/watch?v=gp_D8r-2hwk
- Chronic failures
  - project overruns (time and budget)
  - functionality problems
  - poor performance

Failures 2011

- AXA Rosenburg Group
- Coding error in quantitative investment model
- Company fined $217 million for
  - Withholding information about the error
- Investors made losses which company concealed as losses due to market conditions

Software Crisis today

- HSBC system failure leaves thousands facing bank holiday without pay (August 2015)
  - Caused by error in file sent to BACS
  - 275,000 payments delayed
- United airlines grounded due to failure in booking system (July 2015)
  - Could not check passengers status (including no fly lists)
- Security and trust issues (September 2015)
  - VW emissions fraud

Standish Chaos Report (1995 US)

Total spend on s/w development $ 250 billion
Average cost of project (large company) $ 2,322,000
Average cost of project (medium company) $ 1,331,000
Average cost of project (small company) $ 434,000

31% of projects are cancelled before completion.
52.7 cost 189% of original estimate.
16.2% of projects are completed on time and on budget
For larger companies only 9% are completed on time and on budget

Standish project resolution

- Type 1 Successful
  - Completed on time and on budget with all features
- Type 2 Challenged
  - Over time/budget and incomplete features
- Type 3 Incomplete
  - Project is cancelled

KPMG Report
November 2002 (global)

- 134 listed companies in the UK, US, Africa, Australia and Europe
- 56% written-off at least one software project
- Average loss was €12.5m
- Single biggest loss was €210m
- Causes
  - inadequate planning, poor scope management and poor communication between the IT function and the business

Standish CHAOS report findings

Project overrun reasons

Project Objectives Not Fully Specified	51 percent
Bad Planning and Estimating	48 percent
Technology New to the Organisation	45 percent
Inadequate/No Project Management Methodology	42 percent
Insufficient Senior Staff on the Team	42 percent
Poor Performance by Suppliers Hardware/Software	42 percent
Other-Performance (Efficiency) Problems	42 percent

Standish CHAOS report findings

- Successful projects had:
- User Involvement 15.9%
- Executive Management Support 13.9%
- Clear Statement of Requirements 13.0%
- Proper Planning 9.6%
- Realistic Expectations 8.2%
- Smaller Project Milestones 7%
- Competent Staff 7.2%
- Ownership 5.3%
- Clear Vision & Objectives 2.9%

Reasons for cancel/failed projects

- Incomplete Requirements 13.1%
- Lack of User Involvement 12.4%
- Lack of Resources 10.6%
- Unrealistic Expectations 9.9%
- Lack of Executive Support 9.3%
- Changing Requirements & Specifications 8.7%
- Lack of Planning 8.1%
- Didn't Need It Any Longer 7.5%
- Lack of IT Management 6.2%
- Technology Illiteracy 4.3%

Standish Chaos in perspective

- Did Standish only look for bad news? (article Robert Glass)
- What is the extent of the so called software crisis?
- How many software systems are used in everyday modern life?
- How would you rate their performance?

Chaos report analysed

- How does one define failure?
  - 1 out of 20 features incomplete?
- How does one define overrun?
  - 1 week over the scheduled delivery date?
- Failure of project delivery or estimation technique?
- See The Rise and Fall of the Chaos Report Figures, J. Laurenz Eveleens and Chris Verhoef

Chaos report criticisms

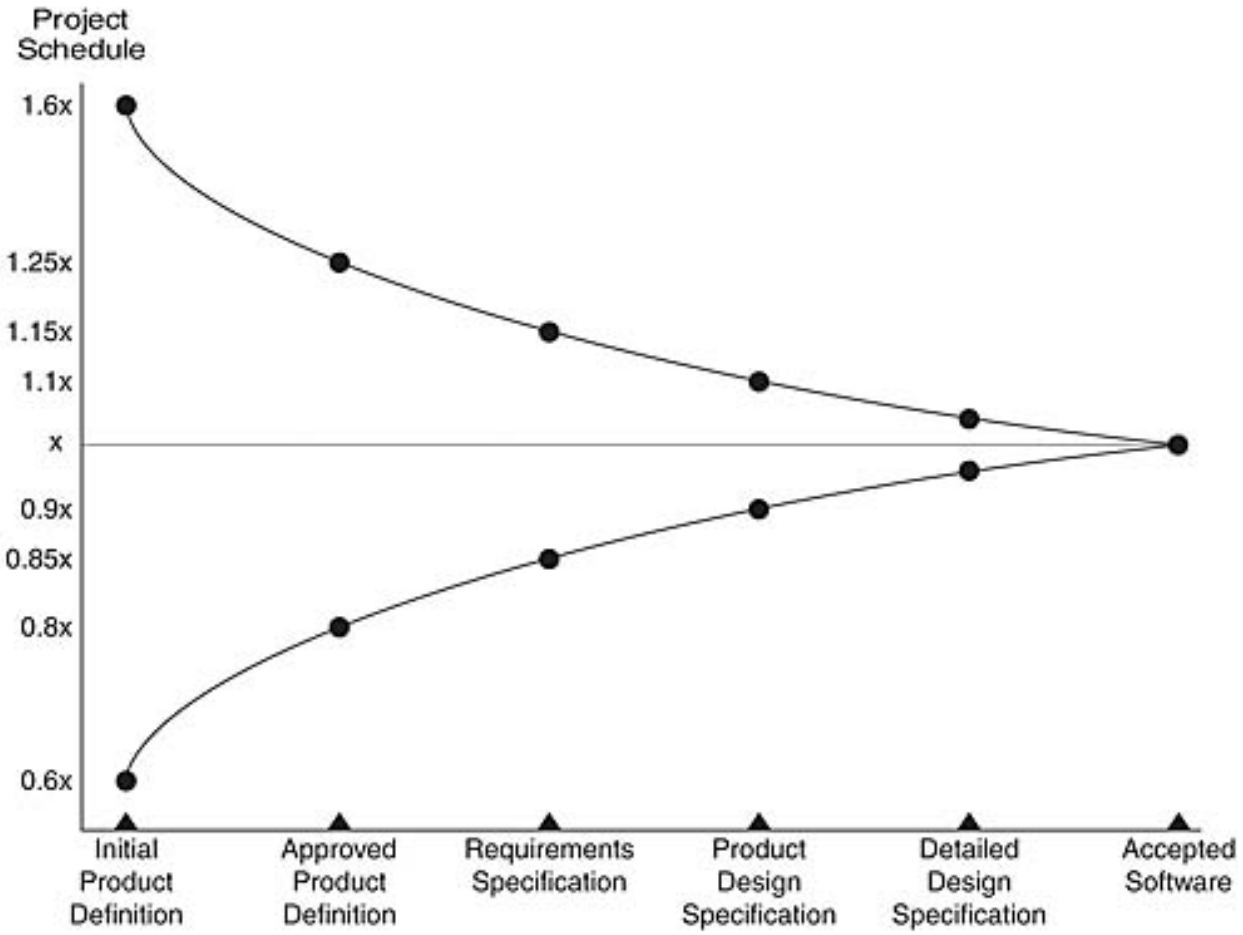
(The Rise and Fall of the Chaos Report Figures)

- Classification of projects incomplete
  - Projects completed within budget and within time
- Raw data not published
- Measuring failure
  - Forcecast/actual
  - f/a <1 (time)
  - f/a >1 (functionality)

Estimation and Chaos report

- Success is measured relative to original estimation
- So...
- Companies sometimes
  - Estimate low timescales
    - Under-resourced, over-promised
  - Estimate high timescales
    - Over-resourced (wasteful)
  - Get it about right

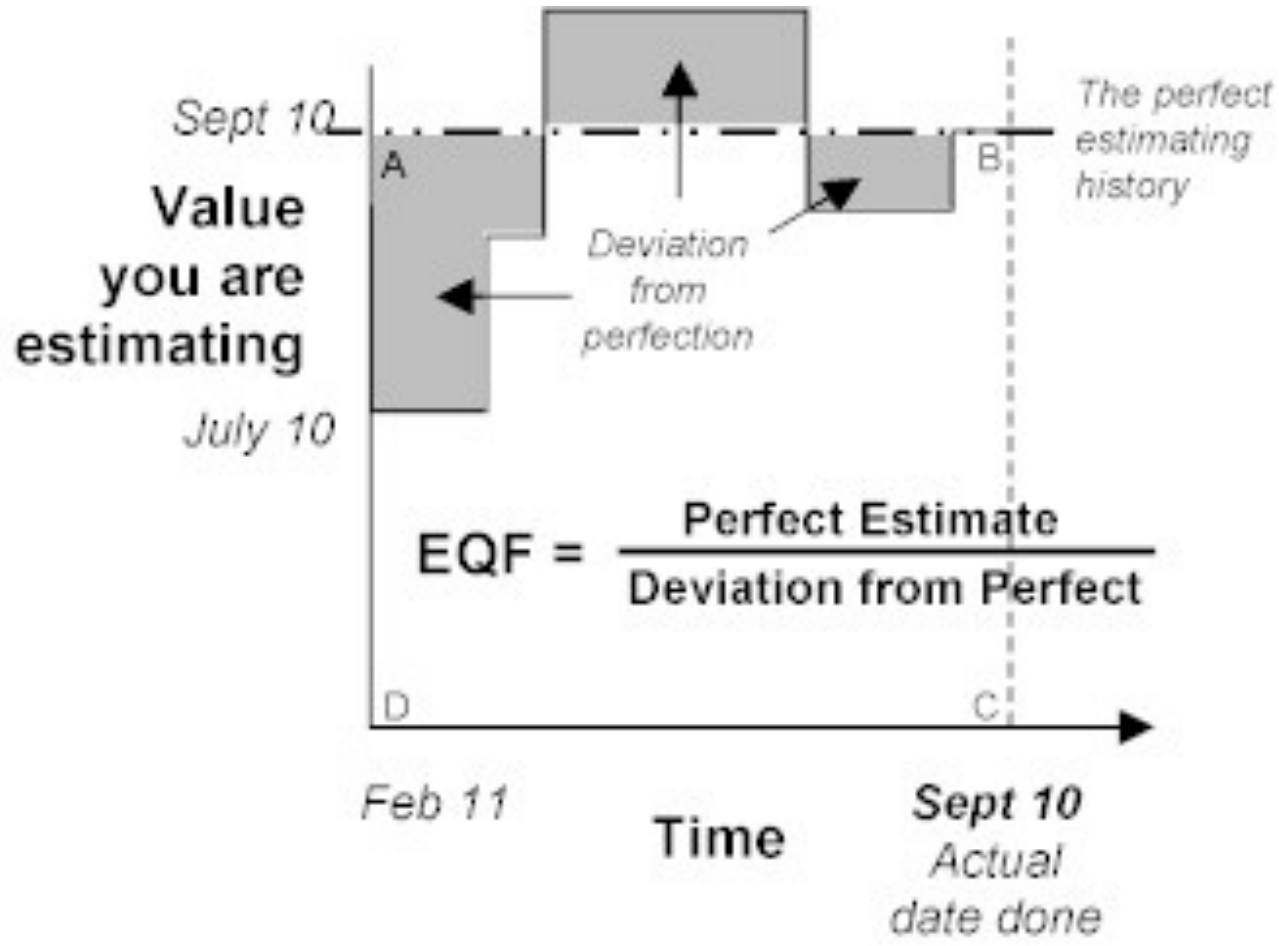
Boem's cone of uncertainty



Project success/failure

- Defined by
  - Context (budget, costings, type of company)
  - Culture
    - Sales driven estimation or development driven estimation
  - Estimation skill
  - Development skill

Estimation quality factor



EQF

- Since there are more than one value you need to average
- EQF = 1 / (Average(Deviation/Actual))

EQF example

- Project completes in 14 weeks
- Estimates
  - 20,10,15,12,12
- Differences
  - 6,4,1,2,2
- EQF
  - Average deviation
    - ((6/14) (4/14) (1/14) (2/14) (2/14)) / 5 = 0.214
    - EQF = 1/Average = 4.67

EQF

- Higher scores show better estimition
- EQF >10 is considered v.good <10% average deviation
- EQF does not show
  - Estimation bias
- Figure for estimation bias can be calculated like EQF but use absolute values

Bias

- In general for bias
  - Bias = Mean(Estimator) – ActualValue
- To normalize the figure (so bias is done as a percentage)

BiasPercentage =

(Mean(Estimator)-ActualValue)/ActualValue

For our example data

- Estimates
  - 20,10,15,12,12
- Average = 13.8
- Bias = (13.8-14)/14
- -0.0143
- Or - 1.43%
- We can see that the estimates are fairly unbiased overall, but slightly optimistic (overall the estimation was under the actual)

Things that effect estimations

- When they are done
  - Early estimations are harder
- Management pressure
  - Sometimes management pressure creates low estimates with low EQF
- Inexperienced developers
  - Inexperienced developers can be overly optimistic/pessimistic
- Lack of design
  - More detailed design makes it easier to estimate
- Quality of the specification

In general

- Very large negative bias
  - Project might have been mean more complex than first thought
  - Project might have changed mid cycle
  - Poorly specified
- Very large positive bias
  - Project has been overestimated due to past experience underestimating
  - Project manager very risk adverse (under pressure from management)

Exercise

- Calculate EQF and bias for the following 3 projects, then draw conclusions
- Project 1
  - Time to complete 20 weeks
  - Estimates 4,4,4,6,7,22,21
- Project 2
  - Time to complete 22 weeks
  - Estimates 18,19,23,24,22
- Project 3
  - Time to complete 50 weeks
  - Estimates 49,50,50,50,50

Rise and Fall of the Chaos Report Figures

- Determination of organisation performance relative to Chaos
- 3 organisations
- Organisation 1
  - 140 projects over 2 years
  - Median f/a of 1.0
  - EQF around 8.5 best in class
  - Used independent consultants to backup their forecasting process
  - Standish success of only 59%

Rise and Fall of the Chaos Report Figures

- Second organisation (X)
  - F/A generally >1 (time) (positive bias)
  - Many projects had surplus budgets
  - Used Standish criteria for determining project success
  - Project managers encouraged to overestimate
  - Average EQF = 0.43
  - Success rate (Chaos) 67%

Rise and Fall of the Chaos Report Figures

- Landmark graphics
  - Forecasts were underestimated
  - So project success rate 6.8%
  - EQF 2.3
- Conclusions
  - Figures on relevant when EQF and bias taken into account
  - Chaos figures from initial report meaningless

Final point on using statistics

- Low birth weight paradox
  - Definitions
    - Babies born under certain weight defined as
      - low birth weight
    - Number of Babies which die in 1st year
      - Infant mortality
  - Paradox
    - Low birth weight babies born to mother who smoke in pregnancy have
      - **LOWER** infant mortality
  - WHY?

Another example

- Harvard University gender bias

Men	Applications	Admitted
Men	8442	44%
Women	4321	35%

Harvard figures broken down

	Men		Women	
A	825	62%	108	**82%**
B	560	63%	25	**68%**
C	325	**37%**	593	34%
D	417	33%	375	**35%**
E	191	**28%**	393	24%
F	272	6%	341	**7%**

What's this to do with software research

- Imagine comparing the performance of
  - A team of software developers
  - 2 software teams
  - 2 software companies
- Are raw figures enough?
- How is performance measured?
- How is size of code measured?

Engineering

"The creative application of scientific principles to design or develop structures, machines, apparatus, or manufacturing processes"

American Engineers' Council for Professional Development

Software Engineering

- “The creative application of scientific principles to design or develop software systems”

Software Engineering History

- 1945 – 1965 Pioneer stage

- 1965 – 1985 Software crisis
  - Research areas
    - Formal methods, high level languages, OO methods
- 1985 – Today No software silver bullet
  - Research areas
    - Methodology and debugging

Software Engineering

"The establishment and use of sound engineering principles in order to obtain economically, software that is reliable and works efficiently on real machines". NATO Science Committee, Fritz Bauer

General Engineering Principles

- Specification
  - What should it do?
  - How should we specify?
- Design
  - How should it do it?
- Manufacture/Implementation
  - Implement the design as product
- Quality control
  - Test/analyse the product
- Modify/enhance
  - Improve the product, fix problems

Software Engineering activities

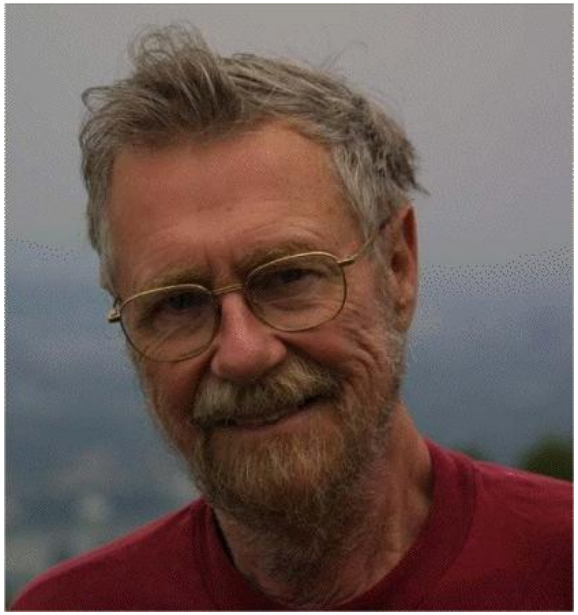
- Requirements analysis
- Software design/implementation
  - Patterns
  - Actor model
  - AOP
- Software testing and analysis
  - Program slicing
- Software Management
  - Scheduling, quality assurance, work flow

Trends going forwards

- A lot of emphasis on Agile
  - Test driven development
  - Good use of software tools
  - Better languages
  - Incremental development
  - SCRUM
  - Improved estimation and project management

Discussion Time

- “… if you carefully read its [Software Engineering] literature and analyse what its devotees actually do, you will discover that software engineering has accepted as its charter "How to program if you cannot."’’



Edsger W. Dijkstra