

# **EBU6304 Software Engineering**

## **Introduction to Software Engineering**

# EBU6304: Software Engineering

## Teaching team:

- Dr Ling Ma(**Module organiser**)
  - [ling.ma@qmul.ac.uk](mailto:ling.ma@qmul.ac.uk)
- Dr Matthew Huntbach:
  - [matthew.huntbach@qmul.ac.uk](mailto:matthew.huntbach@qmul.ac.uk)
- Dr Luca Rossi
  - [luca.rossi@qmul.ac.uk](mailto:luca.rossi@qmul.ac.uk)
- Dr Gokop Goteng
  - [g.l.goteng@qmul.ac.uk](mailto:g.l.goteng@qmul.ac.uk)



# QMPlus and Email

- **Module website:**

- QMPlus →

- <https://qmplus.qmul.ac.uk/course/view.php?id=1101>

- Module Area: EBU6304 – Software Engineering

- Check it regularly, as we could put there information related to e.g. *extra practice exercises*.

- **Email:**

- You are expected to check your email regularly!

- Use your QMUL email or BUPT email ONLY

Emails to the lecturers from other accounts are ignored.

# Questions and feedback

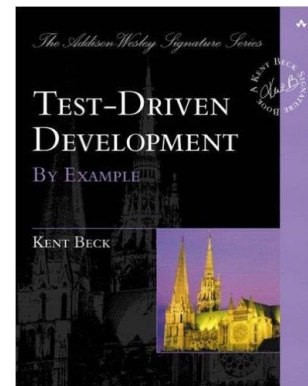
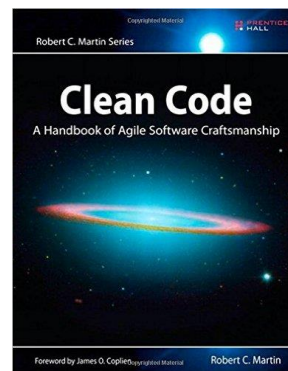
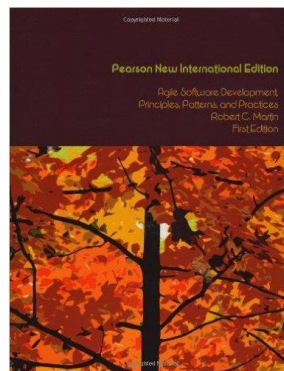
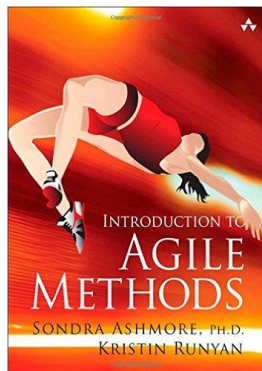
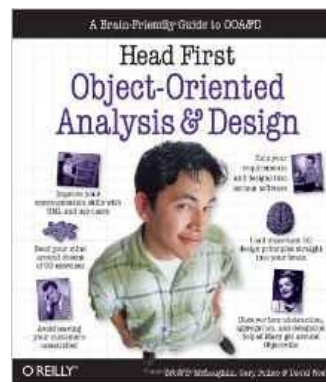
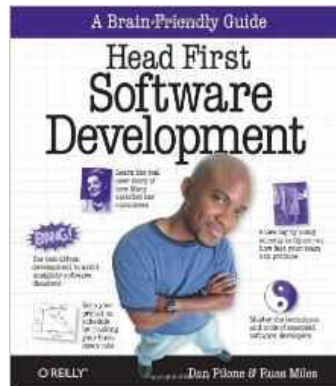
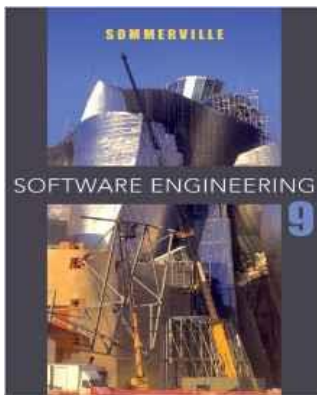
- Message board:
  - Use the “Message Board” forum activity in this course area.
  - For all general questions related to the module
  - Check existing discussions before you post a new question.
  - Notification is sent to QMUL email ONLY

Message board is the primary way of communication for this module

- Feedback:
  - Give lecturers timely feedback is important.

# Recommended Textbooks

See the “**Module Information**” topic of the course website.



# Online resources - ACM, IEEE

- Make use of the QMUL digital library.  
<https://www.library.qmul.ac.uk/>
- ACM Transactions and Digital Library via portal at <http://www.acm.org>, <http://portal.acm.org/portal.cfm>
- ACM Transactions on Software Engineering and Methodology
- IEEE Computer Society <http://www.computer.org>
- IEEE Transactions on Software Engineering
- Software Engineering Journal
- Publications from Wiley, Springer and others

# Module Aims and Objectives

- The module provides:
  - an introduction to **modern** software development techniques necessary to produce high quality software and to manage the production of this software.
  - additional practice in program development.
- The module aims to give each participant:
  - an idea of the necessity of good *software engineering practice* when developing *complex* software systems
  - knowledge of suitable software engineering *techniques* [in particular → practice in applying these techniques]
  - experience of *working in teams* to develop a product to a specification within strict deadlines [in particular → experience of *time-keeping*, which provides valuable experience for the final-year project]

# Topics

- **Week 1**

- Introduction to module & Software Engineering
- Software Processes and Agile Overview
- Requirements

- **Week 2**

- Analysis and Design
- Implementation and Testing
- Project Management

- **Week 3**

- Project Management (continues)
- Design Principles

- **Week 4**

- Design Patterns
- Software Craftsmanship
- Open Source software
- Software Development Tools



# Assessments

- **35% Coursework, made up of:**
  - Individual lab exercise: 5%
  - Group-based project: 30%
- **65% Final Examination**
  - Closed book exam, all compulsory questions.
  - **Duration:** 2 hours

Details of each assessment will be published on QM+ in due course.

# Plagiarism is strictly forbidden!

- What is it?
  - The reproduction of ideas, words or statements of another person without appropriate acknowledgement.
  - Examples:
    - A student knowingly permits another to turn in his/her work.
    - Presenting someone else's work as your own, without giving credit.
- All students must complete their own work and are expected to behave with integrity at all times.
- Plagiarism is strictly forbidden; *there are severe penalties* when detected!
- More information about this at the student handbook.

# EBU6304 – Software Engineering

## Getting started with Software Engineering

- Topics:
  - Overview of software
  - Introduce software engineering and its needs
  - The importance of software engineering

# Questions

**Software = Programs ?**

**Software engineering = Programming ?**

# Software

- Computer programs
- Libraries
- Data (non-executable)
  - Text
  - Digital media
- Documentation
  - System documentation
  - User documentation/ manuals
  - Requirements
  - Design models

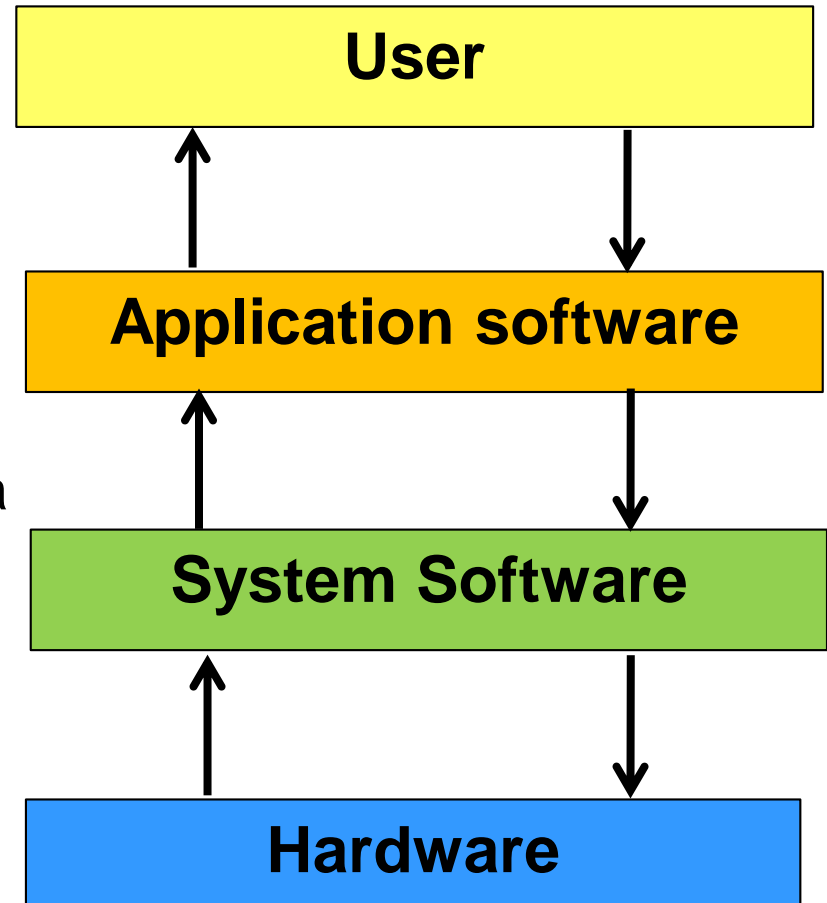
Computer software, or simply software, is a collection of data or computer instructions that tell the computer how to work. This is in contrast to physical hardware, from which the system is built and actually performs the work.

In computer science and software engineering, computer software is all information processed by computer systems, programs and data.

<https://en.wikipedia.org/wiki/Software>

# Software types

- **System** software
  - Operating systems
  - Device drivers
  - Utilities
  - ...
- **Application** software (Perform a specific task)
  - Word processing
  - Image processing
  - Social network
  - ...



# Application Software types

- Stand-alone applications
- Web applications
- Phone App
- Embedded control
- Entertainment
- Modeling and simulation
- Data collection
- ...



# Generic vs Custom

- **Generic** Software
  - Developed for a general market
  - To be sold to a range of different customers
  - **Example:** Microsoft Office, Photoshop
  - Owned and controlled by the development organisation
- **Custom** software
  - Developed for a particular customer, according to their specific needs
  - **Examples:** software used in banks, airlines, embedded systems
  - Owned and controlled by the customer organisation



# Generic vs Custom

- Generic to custom:
  - More and more, software companies are **starting with a generic system** and **customising it to the needs of a particular customer**.
- Examples:
  - University:
    - Moodle – generic
    - QMPLUS – for QMUL
  - Enterprise management: SAP
  - Insurance company
  - Ticket booking
  - e-Commerce

# Good Software?



# Good Software

# Features of “Good” software?



## Results from students

# Good Software

Features of “Good” software?

- Delivers required functionality.
- Dependable
  - Robust, reliable, trustworthy.
- Efficient
  - Good use of resources: computational, user time, development time/cost
- Usable
  - Usable by the users (or systems) it is designed to interact with.

# Good Software

Features of “Good” software?

- Maintainable
  - can evolve to meeting changes in requirements.
- Understandable
- Cost-effective
- Secure/Safe
- ...

# Software Engineering

- "an engineering discipline that is concerned with all aspects of software production"—Ian Sommerville
- "the systematic application of scientific and technological knowledge, methods, and experience to the design, implementation, testing, and documentation of software"—IEEE Systems and software engineering
- "The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software"—IEEE Standard Glossary of Software Engineering Terminology

# Software Engineering

- An engineering discipline
  - Theories, methods, tools, constraints
- Concerned with all aspects for professional software production
- Goal: develop high-quality software
- Systematic and organised approach
- Use appropriate methods, tools and technologies
  - The problem to be solved
  - The development constraints
  - The resources available

# Software Engineer

## Software Engineer



What my friends think I do.



What my mom thinks I do.



What society thinks I do.



What my boss thinks I do.



What I think I do.

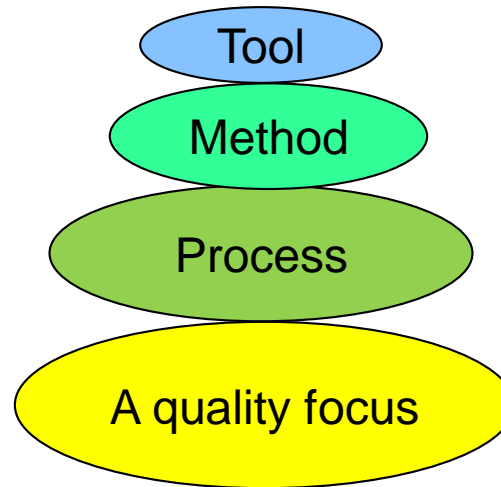


What I actually do.

**Perception** Vs **Fact**.com



# Software engineering layers

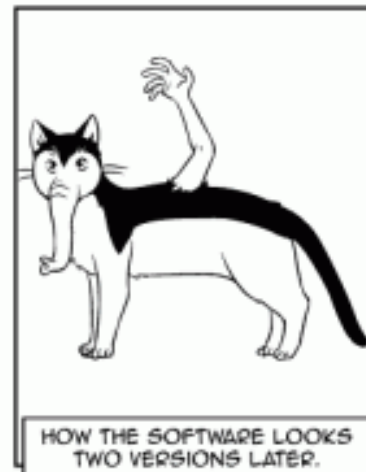
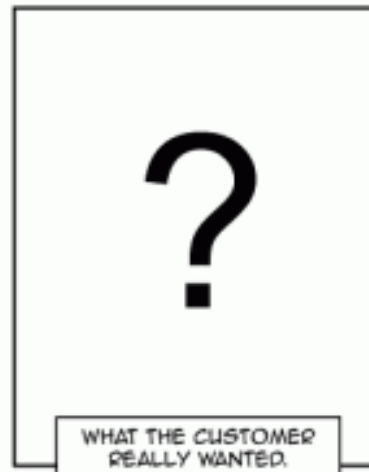
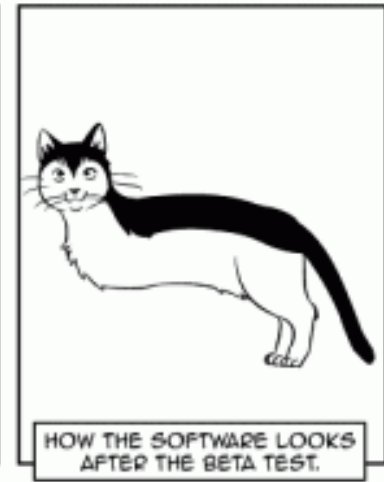
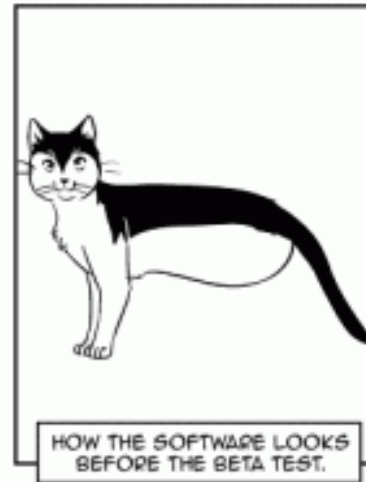
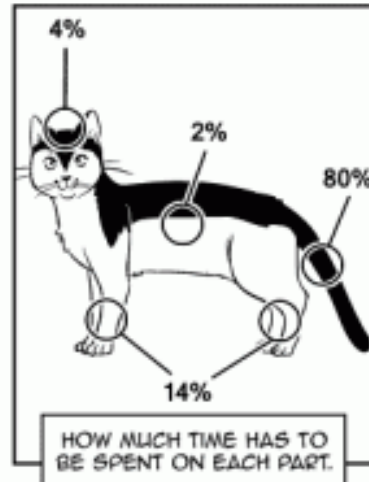


- Software engineering is a **layered technology**
  - Commitment to quality
  - Process: foundation layer
  - Methods: technical layer
  - Tools: support layer

# Why do we need software engineering?

- Questions and issues:
  - Why does it take so long to get software finished?
  - Why are development costs so high?
  - Why can't we find all errors before we give the software to our customers?
  - Why do we continue to have difficulty in measuring progress as software is being developed?
  - Why do we create software that does not fulfil user requirements?

# Richard's guide to software development



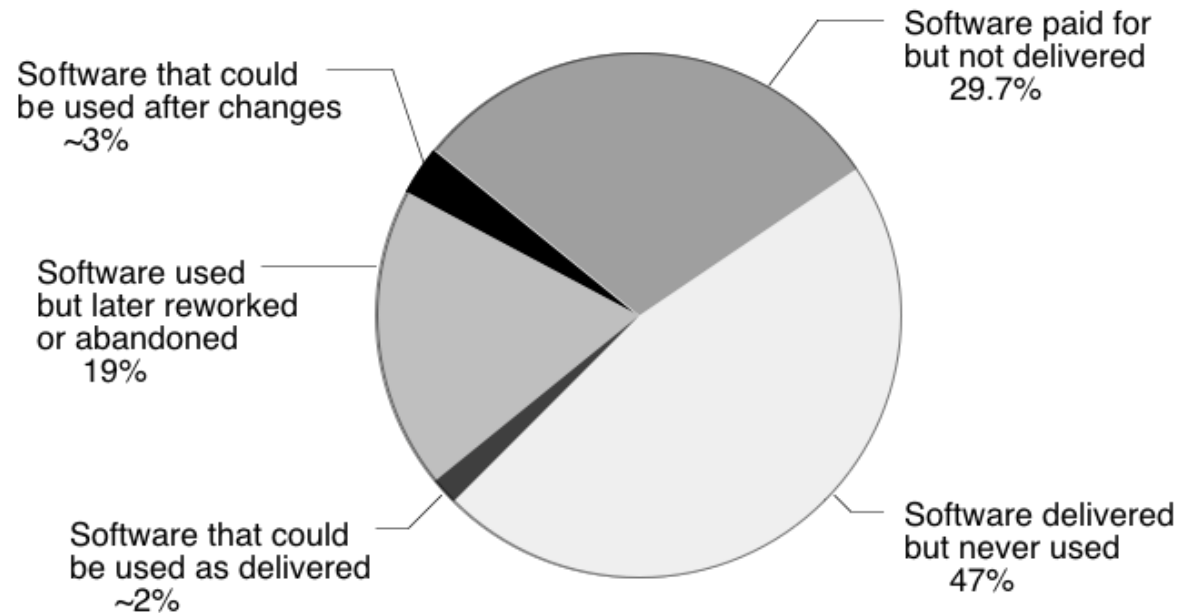
Sandra and Woo by Oliver Knörzer (writer) and Powree (artist) – [www.sandraandwoo.com](http://www.sandraandwoo.com)

# Software failures

- Large scale software development failure
  - Numerous examples of failed or seriously delayed software development projects
  - Or fail to deliver full functionality within time and budget.
  - Or disasters:
    - E.g., NHS IT System, Heathrow T5, Pensions system, Taurus (Stock Exchange), Air Traffic Control, ESA Arienne 5, Patriot Missile System, Therac-25 radiation therapy machine
- Small scale software development failure

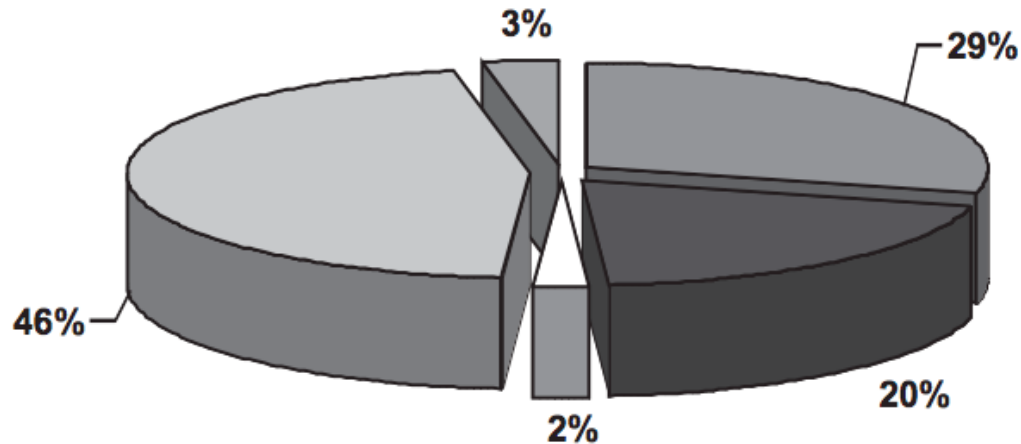
# 1982






## Software Systems Development is Prone to Waste



Year 1982: Nine Contracts Totalling \$6.8 Million

# 1995



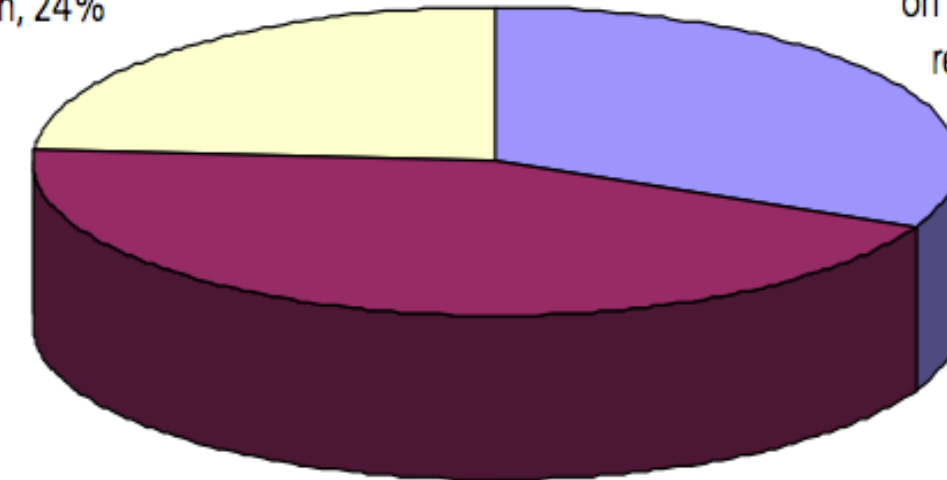
-  **Software paid for, but not delivered - 29%**
-  **Software used, but extensively reworked or abandoned - 20%**
-  **Software used as delivered - 2%**
-  **Software delivered, but not successfully used - 46%**
-  **Software used after changes - 3%**

**Total Software Costs - \$35.7 billion**

# 2009

failed or cancelled before  
completion, 24%

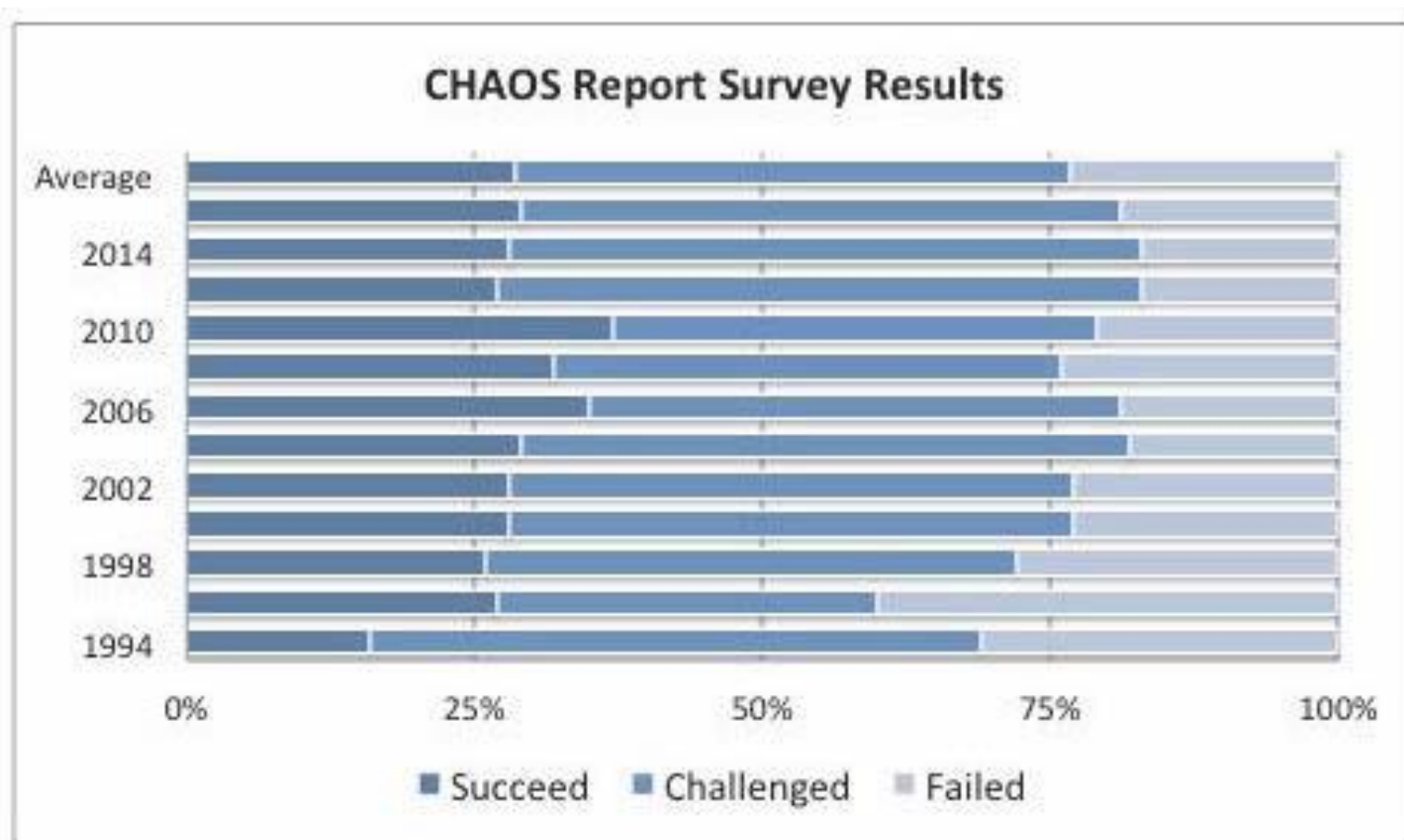
on time, on budget with all  
required features, 32%



late, over budget or without  
required features, 44%

Standish Chaos Report 2009

# 2016





# Present reality

- The reality is mixed
  - Computer Science provides the scientific basis.
  - A growing number of software development approaches are recognised as good engineering practice -Good Design Patterns
  - But many aspects of development are still ad hoc
- Software design is still very difficult
  - Few guiding scientific principles.
  - Few universally applicable methods.
  - Much poor practice & frequent failures.

# General issues that affect software

- Heterogeneity
  - distributed systems, across networks, different types of devices
- Business and social change
  - emerging economies develop, new technologies
- Security and trust
  - it is essential that we can trust that software
- Scale
  - Software has to be developed across a very wide range of scales

# Summary

- Software
  - Types
  - Good software features
- Software engineering
- Software failure
- Software issues

# References and further reading

- **Chapter 1** – “Software Engineering” textbook by Ian Sommerville
- **“Software failure”** examples – see course website

