Brunel University London

Software Evolution

CS2002 Software Development and Management

Giuseppe Destefanis
giuseppe.destefanis@brunel.ac.uk

Topics covered

- Evolution processes
- Legacy systems
- Software maintenance

Software change is inevitable

A key problem for all organizations is implementing and managing change to their existing software systems

Unless change is made properly, the results can be catastrophic

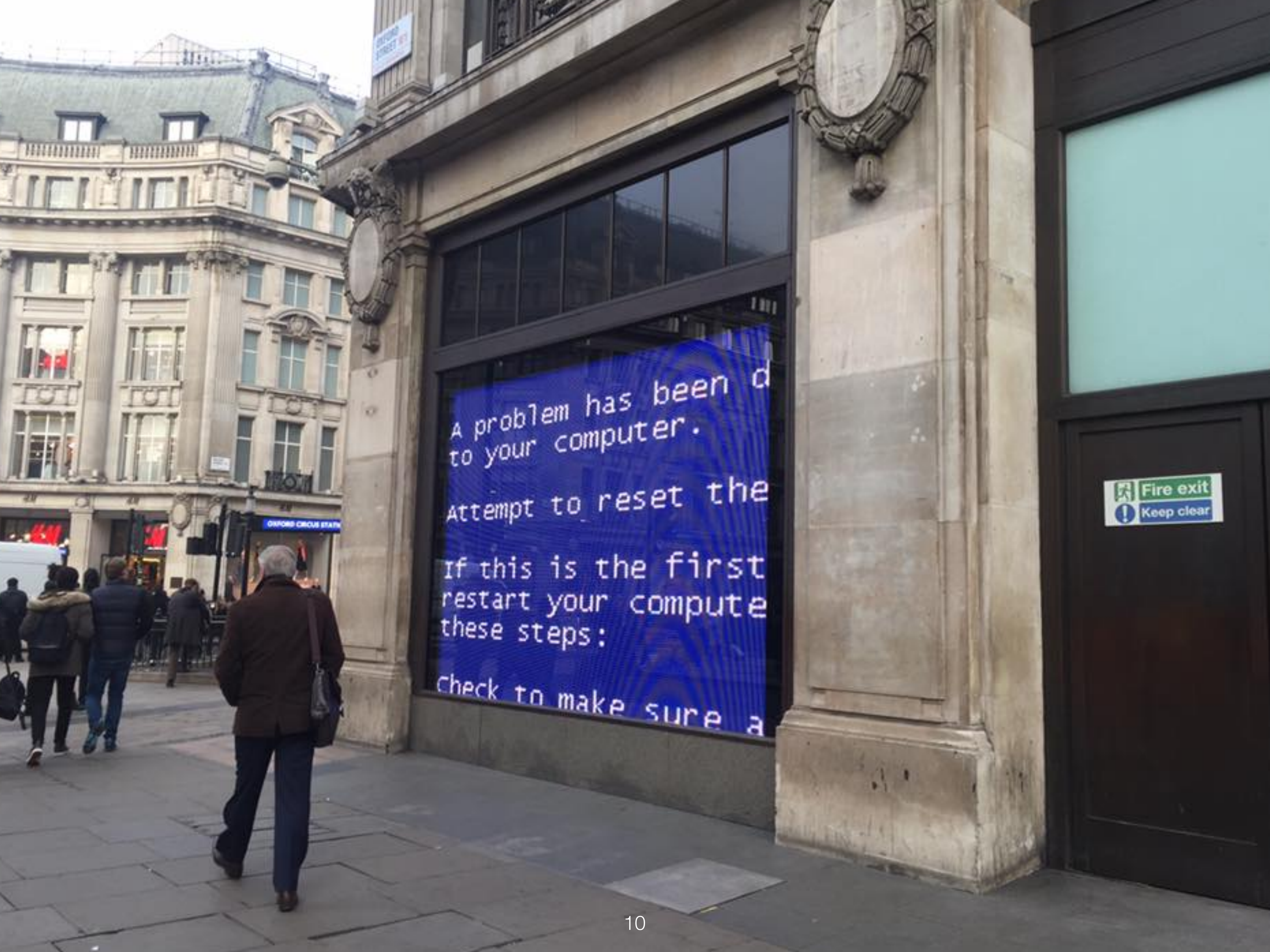
CAE
8. Baza Lotnictwa Transportowego
AIRBUS DEFENCE & SPACE
CAE 7000 Series
C295

Software maintenance

Fault repairs

Changing a system to fix faults/vulnerabilities

(Security fears)



A problem has been d
to your computer.

Attempt to reset the

If this is the first
restart your compute
these steps:

Check to make sure a

Environmental adaptation

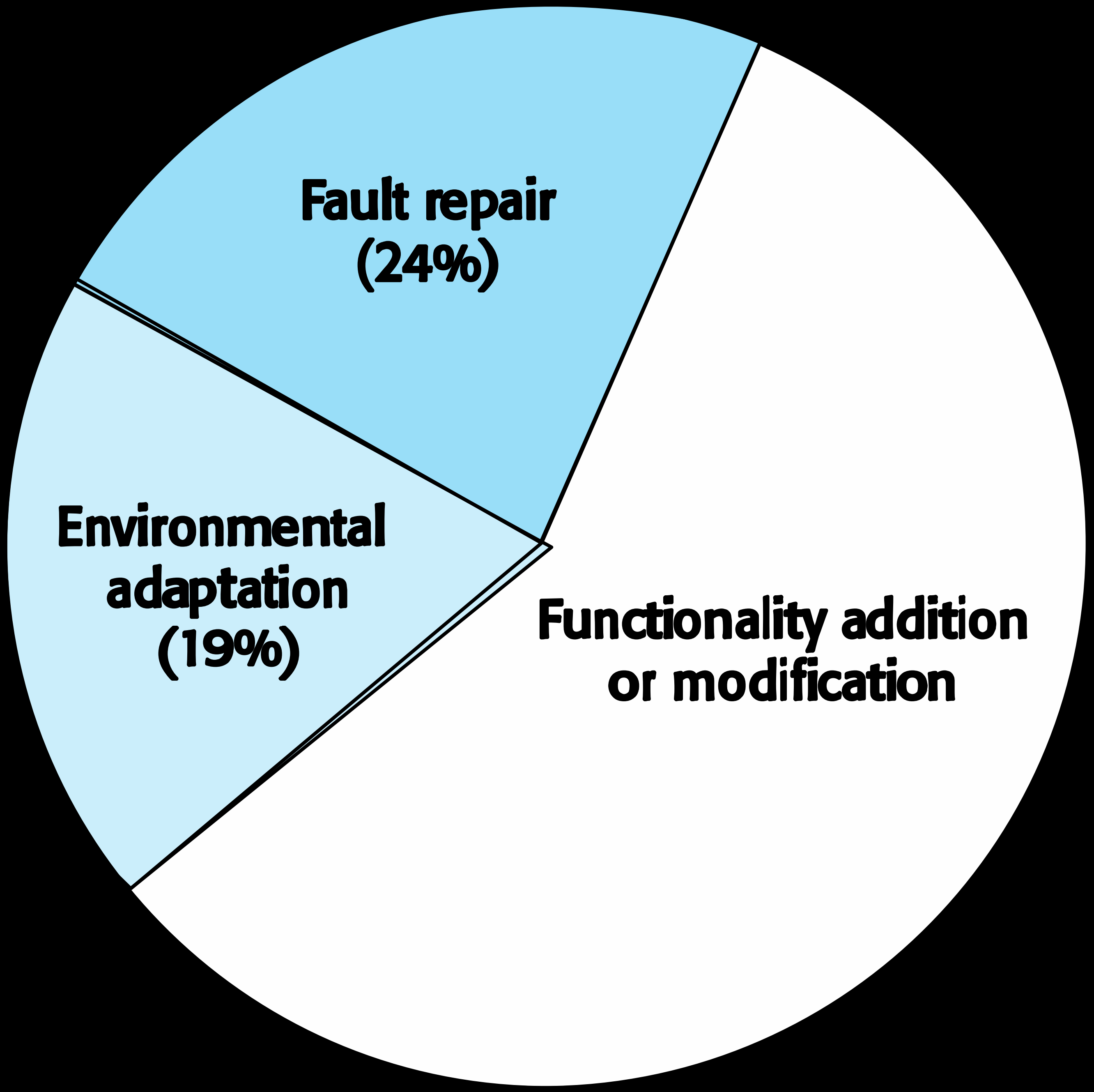
Changing a system so that it operates in a different way from its initial implementation

(New tax laws for example)

Modifying existing functionality

Users asking for features to be changed

Maintenance effort distribution



Program evolution dynamics

Program evolution dynamics is the study of the processes of system change

After major empirical studies, Lehman proposed that there were a number of “laws” which applied to all systems as they evolved

Continuing change

A program that is used in a real-world environment must necessarily change or it becomes progressively less useful in that environment

Increasing complexity

As an evolving program changes, its structure tends to become more complex. Extra resources must be devoted to preserving and simplifying the structure

Continuing growth

The functionality offered by systems has to continually increase to maintain user satisfaction

Declining quality

The quality of systems will decline unless they are modified to reflect changes in their operational environment

Feedback system

Evolution processes incorporate multiagent, multiloop feedback systems and you have to treat them as feedback systems to achieve significant product improvement

Legacy system replacement

```
                Manzanita Software Systems

              ((( The Business Accountant )))
                           for
                  HUFFMAN [illegible] INC

  Available options are:

        ( 1) - General Ledger System
        ( 2) - Accounts Payable System
        ( 3) - Accounts Receivable System
        ( 4) - Inventory Control System
        ( 5) - Payroll
        ( 6) - Labels Plus
        ( 7) - System Utilities
        ( 8) - System Setup and Configuration
        ( 9) - Change Today's Date
        (10) - System Status Report

  Enter option number (1 to 10) or <RETURN> only to quit: █

          Copyright 1984 by Manzanita Software Systems
```

Legacy system replacement is risky

- Lack of complete original system specification
  - No means of reusing the old designs

Legacy system replacement is risky

- Tight integration of system and business processes
  - You have to model the business – and that is difficult

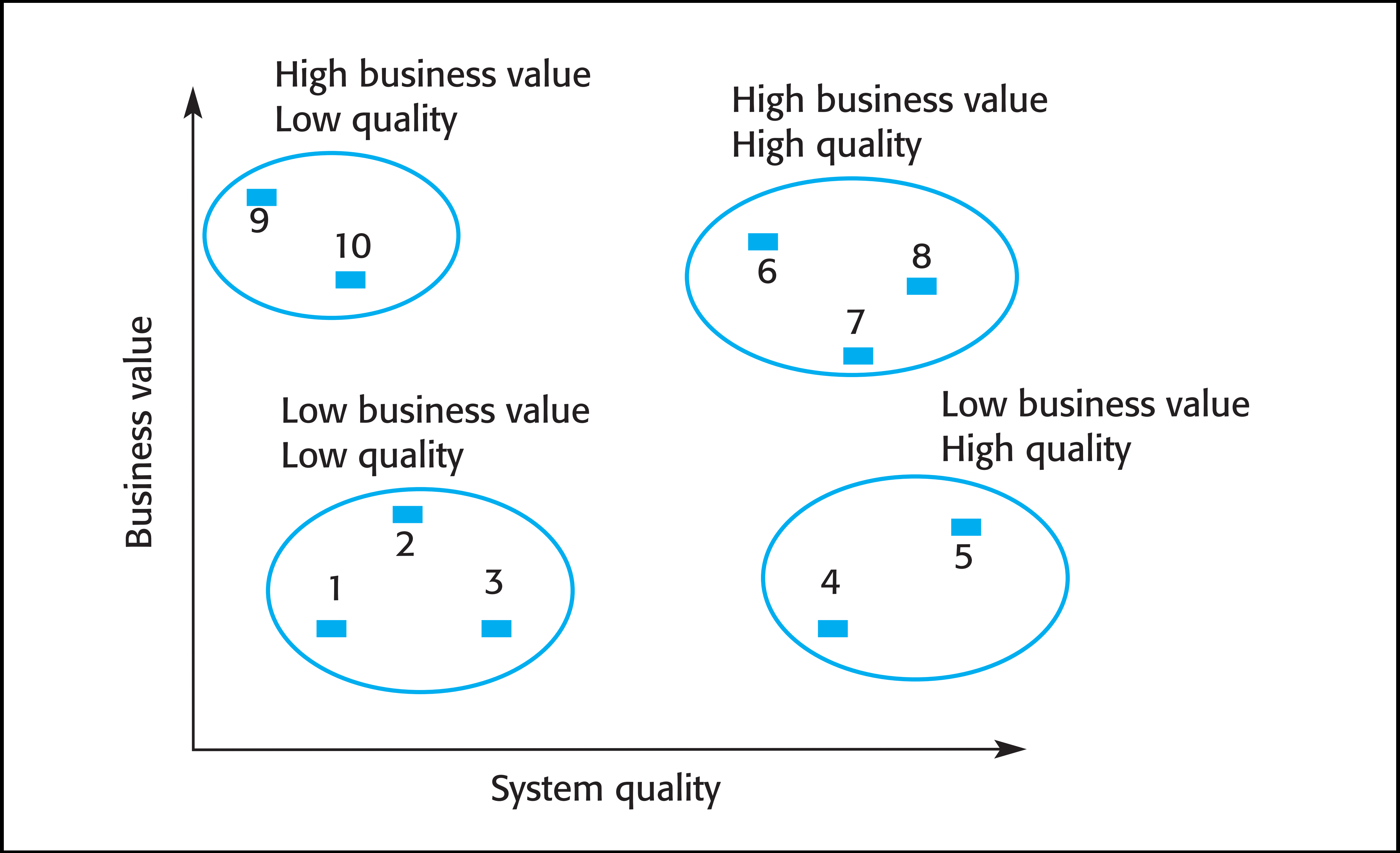
Legacy system replacement is risky

- Undocumented business rules embedded in the legacy system
    - Poor upkeep of documentation

Legacy system replacement is risky

- New software development is always late and over budget
    - Replacement will always be over budget and late

Legacy system assessment



Legacy system categories

- **Low quality, low business value**
  - These systems should be scrapped
- **Low quality, high business value**
  - These make an important business contribution but are expensive to maintain
  - Should be re-engineered or replaced if a suitable system is available
- **High quality, low business value**
  - Scrap completely or maintain (?)
- **High quality, high business value**
  - Continue in operation using normal system maintenance

Reengineering cost factors

- **The quality of the software to be reengineered**
    - Can software metrics help here to measure quality?
- **The tool support available for reengineering**
    - Developers are always moaning about tools
    - Refactoring tools are poor
- **The availability of expert staff for reengineering**
    - This can be a problem with old systems based on technology that is no longer widely used
- **Time (costs money)**
    - The cost of one developer per day
- Technical debt

Factors that affect system lifetimes

Investment cost

The costs of a systems engineering project may be tens or even hundreds of millions of pounds

Replacement cost

The cost of replacing a large system is very high.

Replacing an existing system can only be justified if this leads to significant cost savings over the existing system

Return on investment

If a fixed budget is available for systems engineering, spending this on new systems in some other area of the business may lead to a higher return on investment than replacing an existing system

Risks of change

The danger with a new system is that things can go wrong in the hardware, software and operational processes.

The potential costs of these problems for the business may be so high that they cannot take the risk of system replacement

System dependencies

Other systems may depend on a system and making changes to these other systems to accommodate a replacement system may be impractical

Refactoring

Refactoring is the process of making improvements to a program to slow down degradation through change

Refactoring

- You can think of refactoring as ‘preventative maintenance’ that reduces the problems of future change
- When you refactor a program, you should not add functionality but rather concentrate on program improvement
- Example: Move method (reduces coupling)

‘Bad smells’ in program code

- **Duplicate code**
    - The same or very similar code may be included at different places in a program
    - This can be removed and implemented as a single method or function that is called as required
- **Long method**
    - If a method is too long, it should be redesigned as a number of shorter methods
- **Large class**
    - A class has too many methods
    - Coupling becomes an issue

System measurement

- You may collect quantitative data to make an assessment of the quality of the application system
  - **The number of system change requests**
    - The higher this accumulated value, the lower the quality of the system
  - **The number of faults over a period of time**
    - Higher the number of faults, the lower the quality of the system
  - **McCabe’s Cyclomatic Complexity**
    - Too much complexity reflects a poor quality system

Reading

- Chapter 9 of Somerville
- Parts of Chapter 19