Chapter 14 – Software Evolution

Lecture 2

Software maintenance

- Modifying a program after it has been put into use.
- The term is mostly used for changing custom software. Generic software products are said to evolve to create new versions.
- Maintenance does not normally involve major changes to the system's architecture.
- Changes made to the software may be simple changes to correct coding errors, more extensive changes to correct design errors
- Changes are implemented by modifying existing components and adding new components to the system.

Types of maintenance

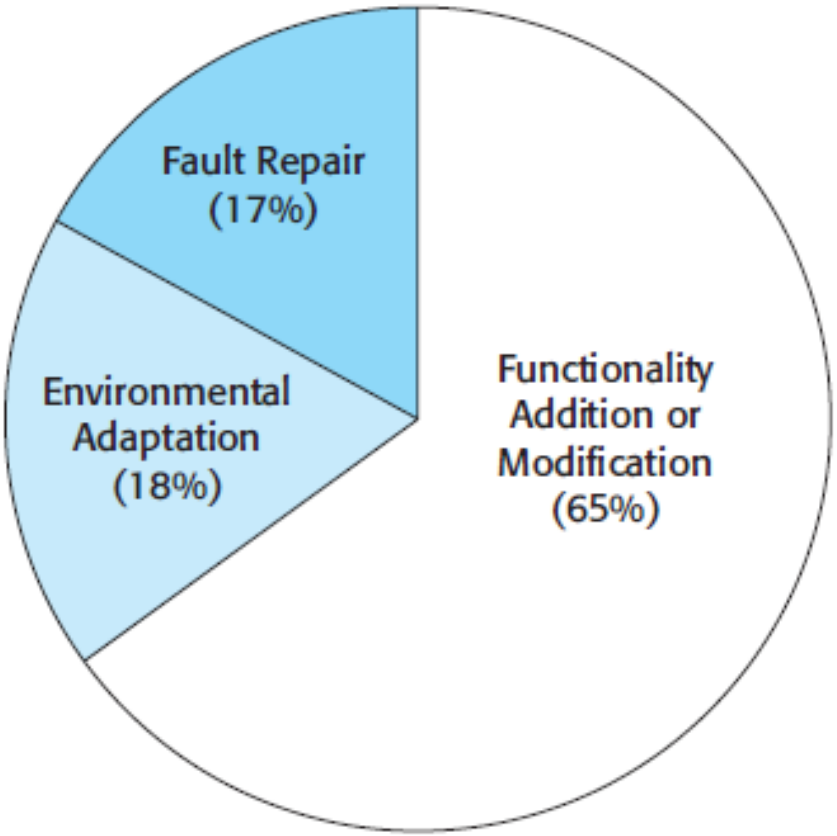
✧ Maintenance to repair software faults

- Changing a system to correct deficiencies in the way meets its requirements.
- Coding errors are usually relatively cheap to correct; design errors are more expensive as they may involve rewriting several program components.
- Requirements errors are the most expensive to repair because of the extensive system redesign which may be necessary.

Types of maintenance

- Maintenance to adapt software to a different operating environment
  - required when some aspect of the system's environment such as the hardware, the platform operating system, or other support software changes.
  - The application system must be modified to adapt it to cope with these environmental changes.
- Maintenance to add to or modify the system's functionality
  - Modifying the system to satisfy new requirements.
  - The scale of the changes required to the software is often much greater than for the other types of maintenance.

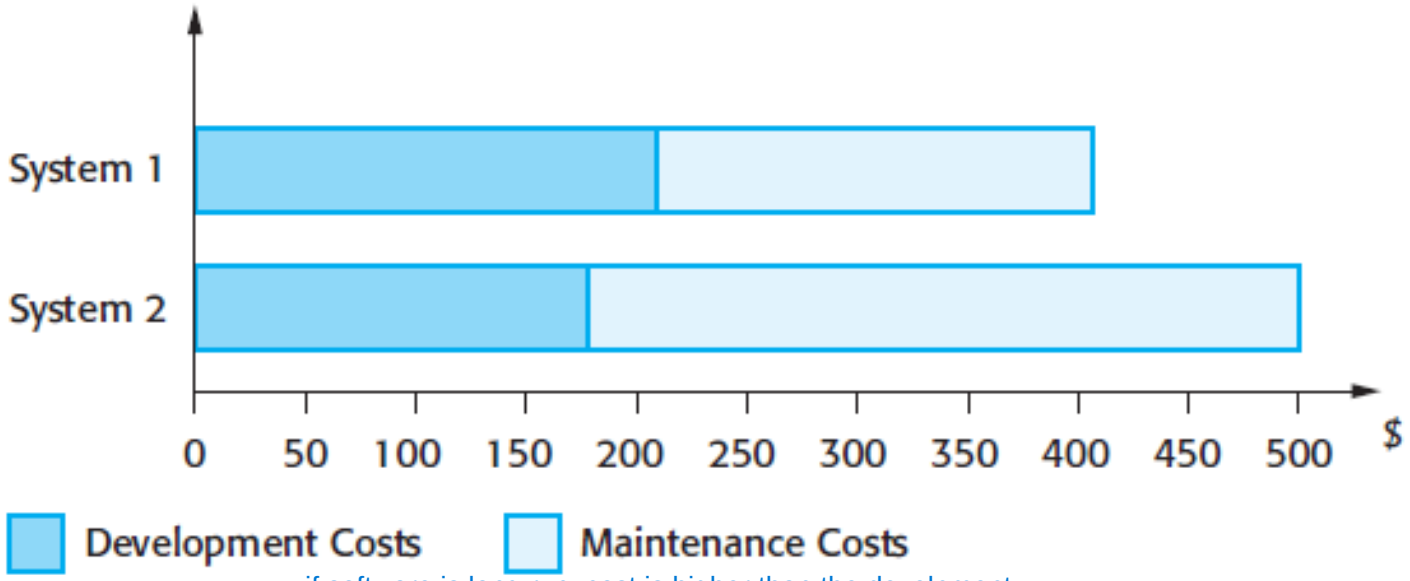
Figure 9.8 Maintenance effort distribution



Maintenance costs

- Usually greater than development costs (2* to 100* depending on the application).
- Affected by both technical and non-technical factors.
- Increases as software is maintained. Maintenance corrupts the software structure so makes further maintenance more difficult.
- Ageing software can have high support costs (e.g. old languages, compilers etc.).

Figure 9.9 Development and maintenance costs



if software is long run, cost is higher than the develpment

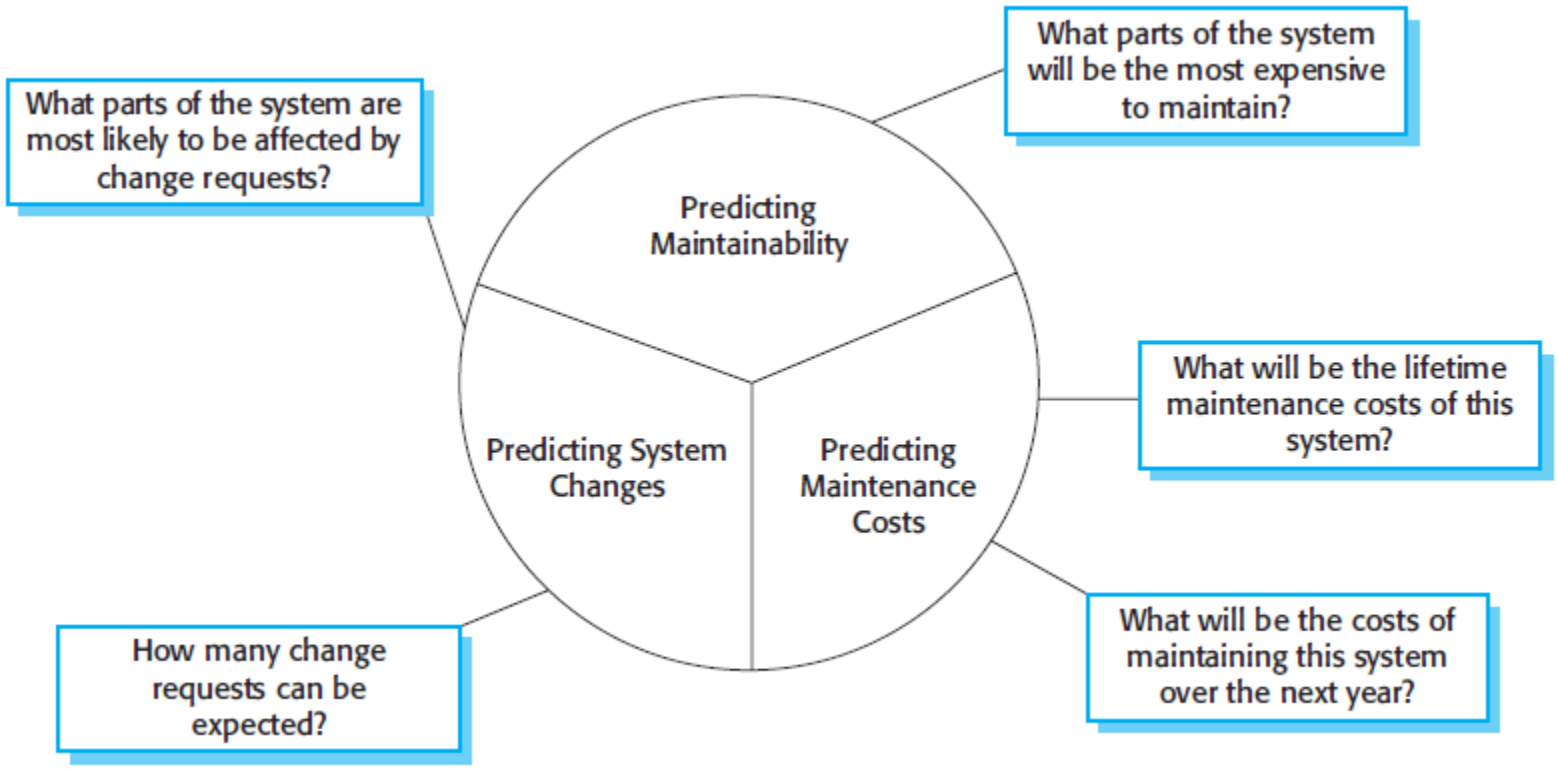
Maintenance cost factors

- ✧ Team stability
  - Maintenance costs are reduced if the same staff are involved with them for some time.
- ✧ Contractual responsibility
  - The developers of a system may have no contractual responsibility for maintenance so there is no incentive to design for future change.
- ✧ Staff skills
  - Maintenance staff are often inexperienced and have limited domain knowledge.
- ✧ Program age and structure
  - As programs age, their structure is degraded and they become harder to understand and change.

Maintenance prediction

- Maintenance prediction is concerned with assessing which parts of the system may cause problems and have high maintenance costs
  - Change acceptance depends on the maintainability of the components affected by the change;
  - Implementing changes degrades the system and reduces its maintainability;
  - Maintenance costs depend on the number of changes and costs of change depend on maintainability.

Maintenance prediction



Change prediction

- Predicting the number of changes requires and understanding of the relationships between a system and its environment.
- Tightly coupled systems require changes whenever the environment is changed.
- Factors influencing this relationship are
  - Number and complexity of system interfaces;
  - Number of inherently volatile system requirements;
  - The business processes where the system is used.

Complexity metrics

- Predictions of maintainability can be made by assessing the complexity of system components.
- Studies have shown that most maintenance effort is spent on a relatively small number of system components.
- Complexity depends on
  - Complexity of control structures;
  - Complexity of data structures;
  - Object, method (procedure) and module size.

System re-engineering

- Many systems, especially older legacy systems, are difficult to understand and change.
- A legacy system is outdated computing software and/or hardware that is still in use.
- To make legacy software systems easier to maintain, you can reengineer these systems to improve their structure and understandability
- Re-structuring or re-writing part or all of a legacy system without changing its functionality.
- Applicable where some but not all sub-systems of a larger system require frequent maintenance.

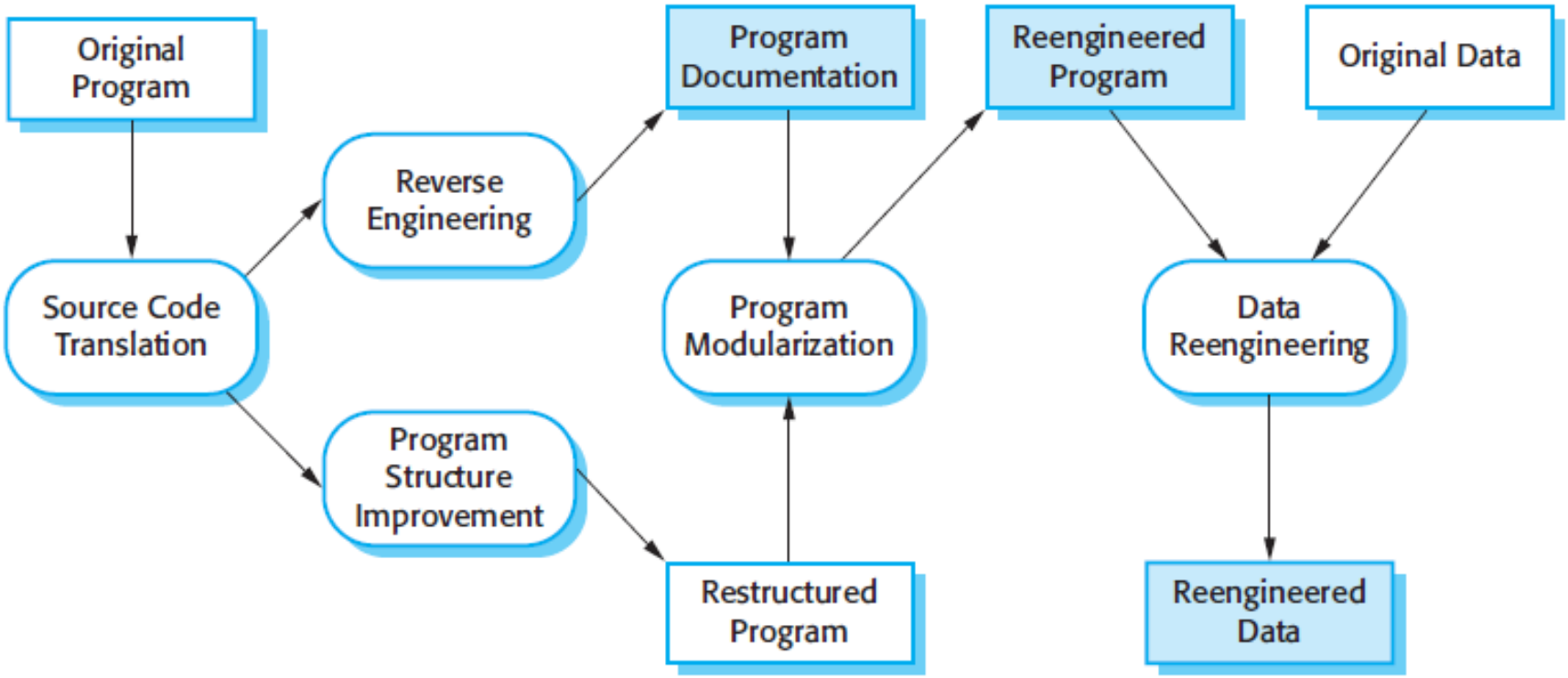
System re-engineering

- Re-engineering involves adding effort to make them easier to maintain. The system may be re-structured and re-documented.
- Reengineering may involve redocumenting the system, refactoring the system architecture, translating programs to a modern programming language, and modifying and updating the structure and values of the system’s data.
- The functionality of the software is not changed and, normally, you should try to avoid making major changes to the system architecture.

Advantages of reengineering

- ✧ Reduced risk
  - There is a high risk in new software development. There may be development problems, staffing problems and specification problems.
- ✧ Reduced cost
  - The cost of re-engineering is often significantly less than the costs of developing new software.

The reengineering process



Reengineering process activities

- ✧ Source code translation
  - Convert code to a new language.
- ✧ Reverse engineering
  - Analyse the program to understand it;
  - This helps to document its organization and functionality. Again, this process is usually completely automated.
- ✧ Program structure improvement
  - The control structure of the program is analyzed and modified to make it easier to read and understand.
  - Restructure automatically for understandability;

Reengineering process activities

- ✧ Program modularization
  - Reorganize the program structure;
  - Related parts of the program are grouped together and, where appropriate, redundancy is removed
  - This is a manual process.
- ✧ Data reengineering
  - The data processed by the program is changed to reflect program changes.
  - This may mean redefining database schemas and converting existing databases to the new structure.
  - Clean-up and restructure system data.

Reengineering cost factors

- ✧ The quality of the software to be reengineered.
- ✧ The tool support available for reengineering.
- ✧ The extent of the data conversion which is required.
- ✧ The availability of expert staff for reengineering.
  - This can be a problem with old systems based on technology that is no longer widely used.

Preventative maintenance by refactoring

- Refactoring is the process of making improvements to a program to slow down degradation through change.
- You can think of refactoring as ‘preventative maintenance’ that reduces the problems of future change.
- Refactoring involves modifying a program to improve its structure, reduce its complexity or make it easier to understand.
- When you refactor a program, you should not add functionality but rather concentrate on program improvement.

Refactoring and reengineering

- Re-engineering takes place after a system has been maintained for some time and maintenance costs are increasing. You use automated tools to process and re-engineer a legacy system to create a new system that is more maintainable.
- Refactoring is a continuous process of improvement throughout the development and evolution process. It is intended to avoid the structure and code degradation that increases the costs and difficulties of maintaining a system.

‘Bad smells’ in program code

Examples of bad smells that can be improved through refactoring include:

- ✧ 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 methods
  - If a method is too long, it should be redesigned as a number of shorter methods.
- ✧ Switch (case) statements
  - These often involve duplication, where the switch depends on the type of a value. The switch statements may be scattered around a program. In object-oriented languages, you can often use polymorphism to achieve the same thing.

‘Bad smells’ in program code

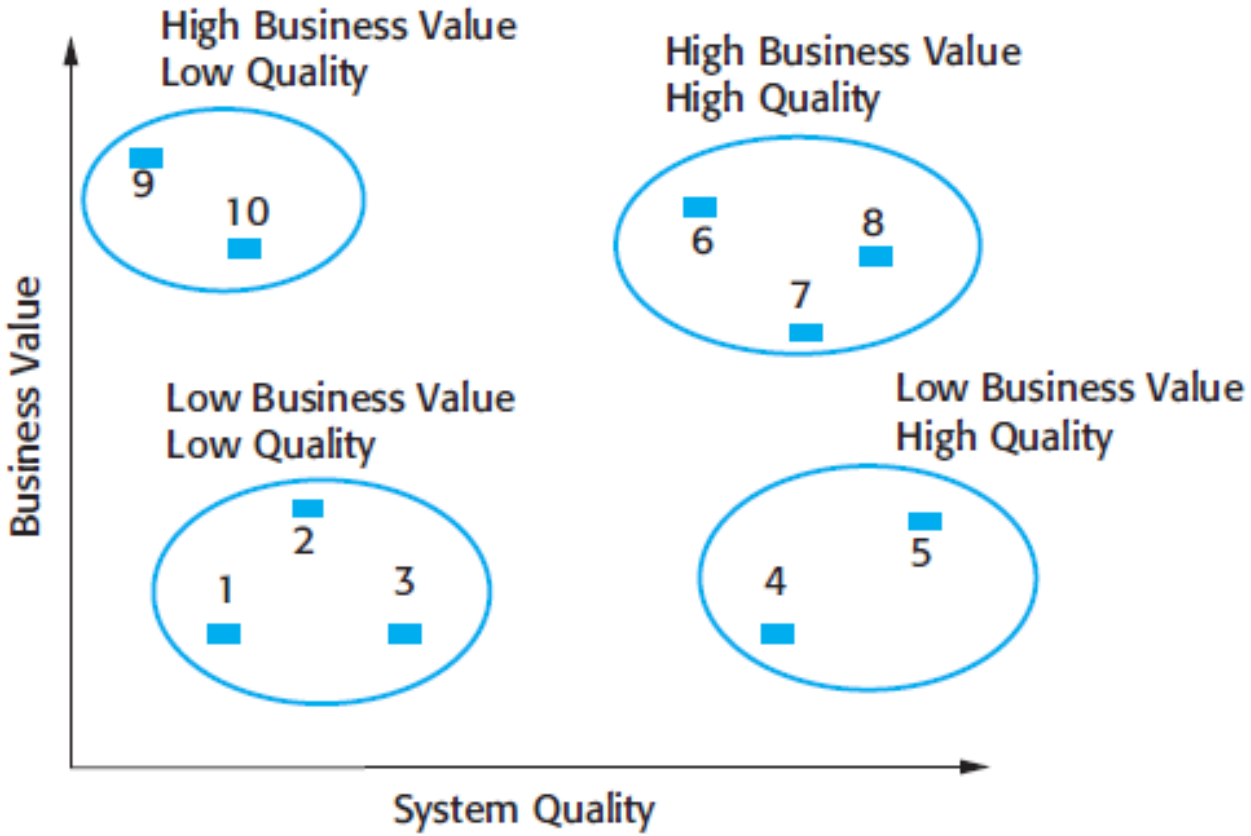
✧ Data clumping

- Data clumps occur when the same group of data items (fields in classes, parameters in methods) re-occur in several places in a program. These can often be replaced with an object that encapsulates all the data.

Legacy system management

- Organisations that rely on legacy systems must choose a strategy for evolving these systems
  - Scrap the system completely and modify business processes so that it is no longer required;
  - Continue maintaining the system;
  - Transform the system by re-engineering to improve its maintainability;
  - Replace the system with a new system.
- The strategy chosen should depend on the system quality and its business value.

Figure 9.13 An example of a 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
  - Replace with COTS, scrap completely or maintain.
- High-quality, high business value
  - Continue in operation using normal system maintenance.

Key points

- ✧ There are 3 types of software maintenance, namely bug fixing, modifying software to work in a new environment, and implementing new or changed requirements.
- ✧ Software re-engineering is concerned with re-structuring and re-documenting software to make it easier to understand and change.
- ✧ Refactoring, making program changes that preserve functionality, is a form of preventative maintenance.

Classwork Exercise

As a software project manager in a company that specializes in the development of software for the offshore oil industry, you have been given the task of discovering the factors that affect the maintainability of the systems developed by your company. Suggest how you might set up a program to analyze the maintenance process and discover appropriate maintainability metrics for your company.

Classwork Exercise (Revisited)

Explain why the environment in which a computer-based system is installed may have unanticipated effects on the system that lead to system failure. Illustrate your answer with a different example from that used in this chapter.

Reference: Software Engineering by Ian Sommerville, Pearson Education, Inc., publishing as Addison-Wesley