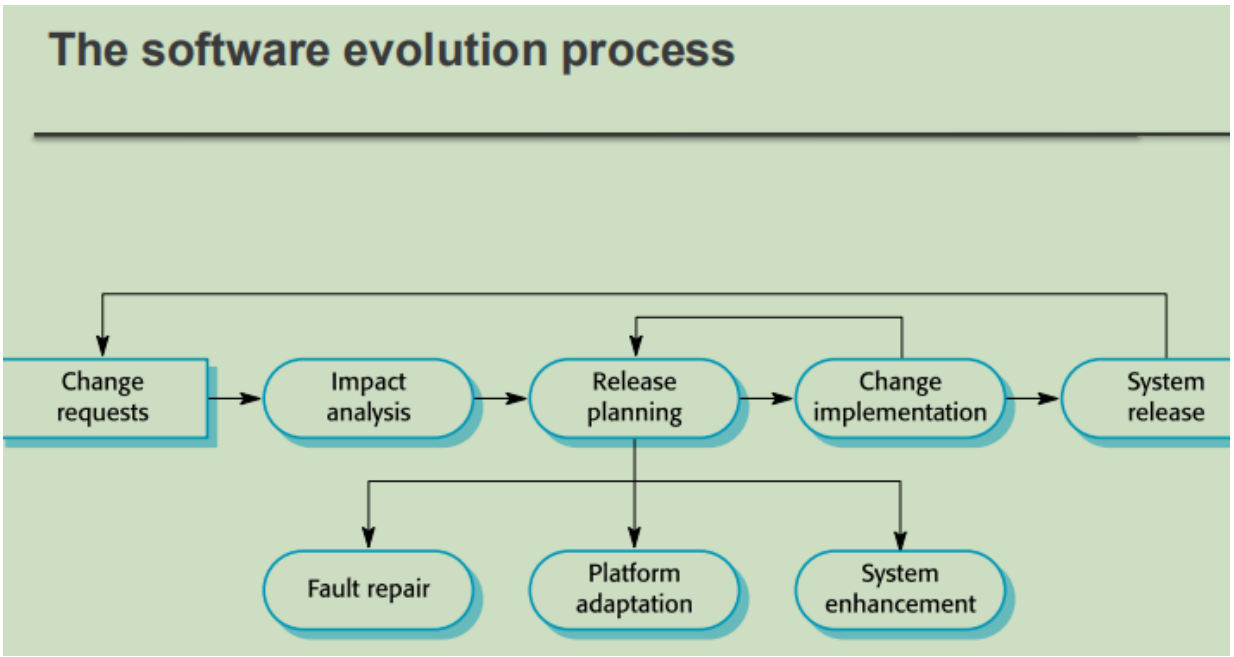
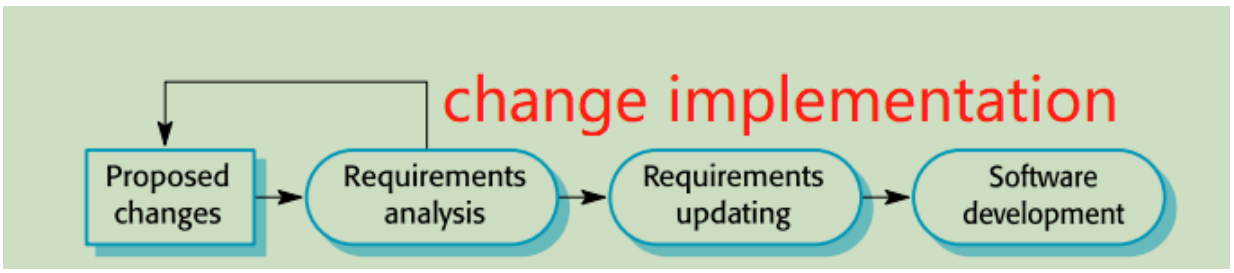
L10 Software Evolution

- Key problem: implementing and managing change to their existing software systems
- Evolution: operational use
- Servicing: useful but fix bus. No new functionality
- Phase-out: may still useful but no further changes are made

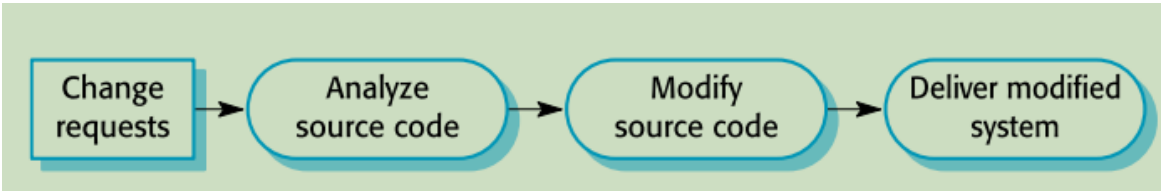
Evolution processes

- change identification and evolution continues throughout the system lifetime



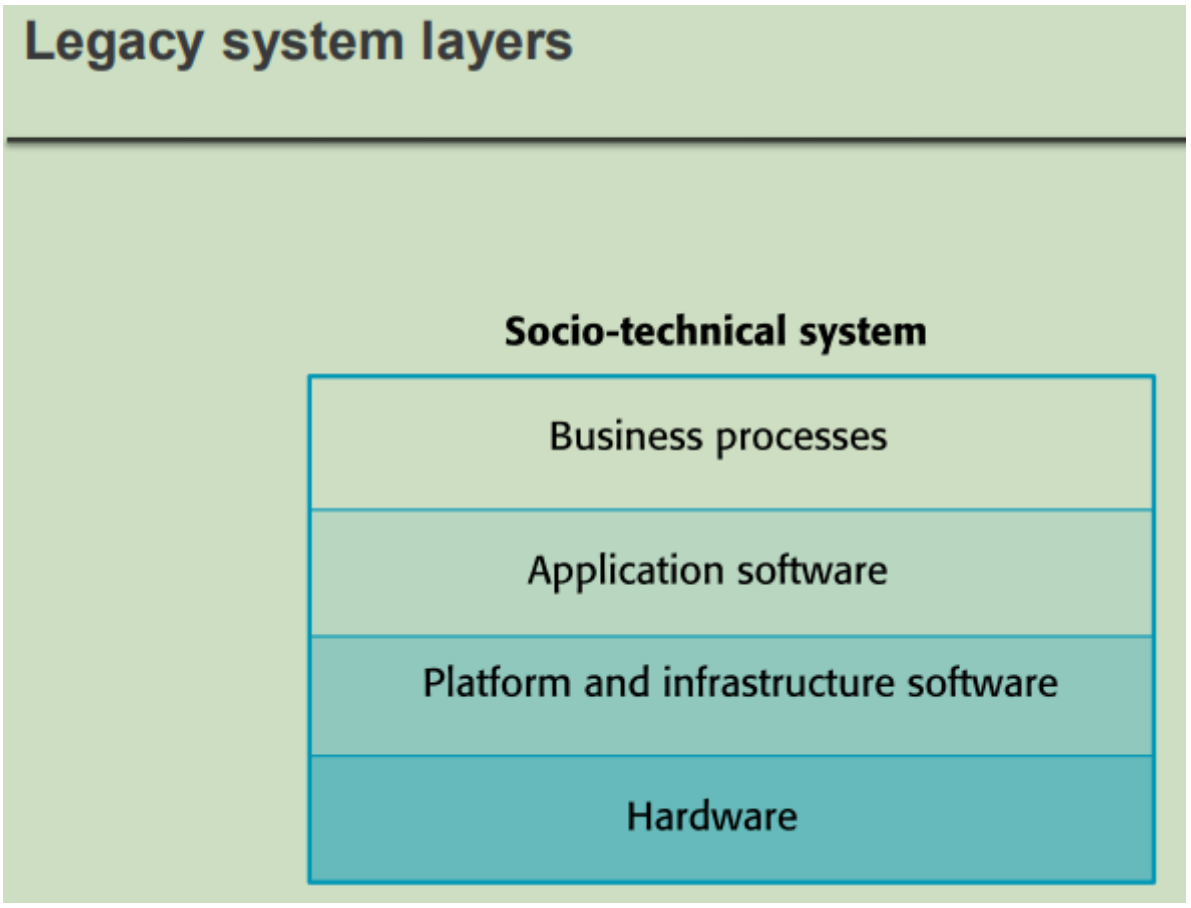


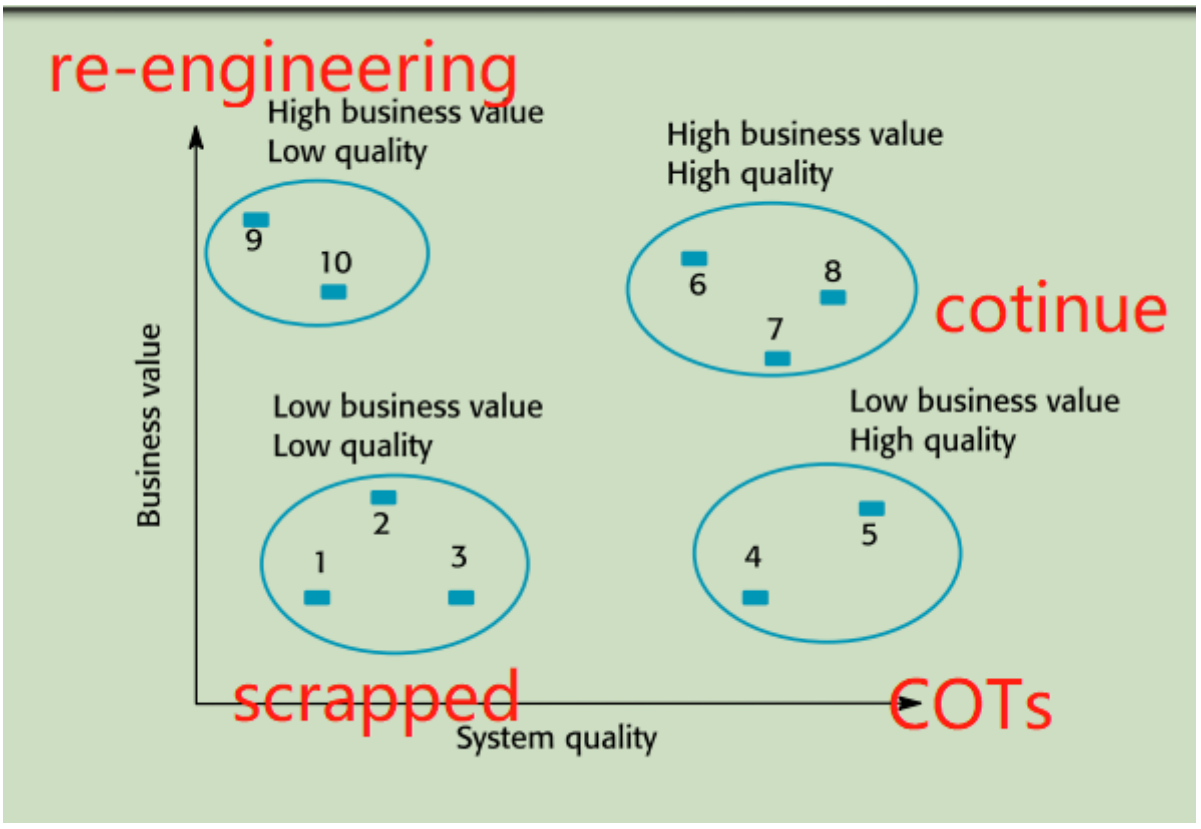
- program understanding; developers change
- Urgent change requests: without going through all stages
  - serious fault
  - changes to the system's environment have unexpected effects
  - business changes , rapd response
  - need:



Legacy systems

- Legacy systems are older systems that rely on languages and technology that are no longer used for new system development
- Not just software but broader: hardware, software, libraries, other supporting software and business processes
- Components
  - *system hardware*
  - *support software*
  - *application software*: application system that provides the bussiness services is usually made up of a number of application programs
  - *application data*: These are data that are processed by the application system. They may be inconsistent, duplicated or held in different databases
  - *business processes*
  - *business pplicies and rules*





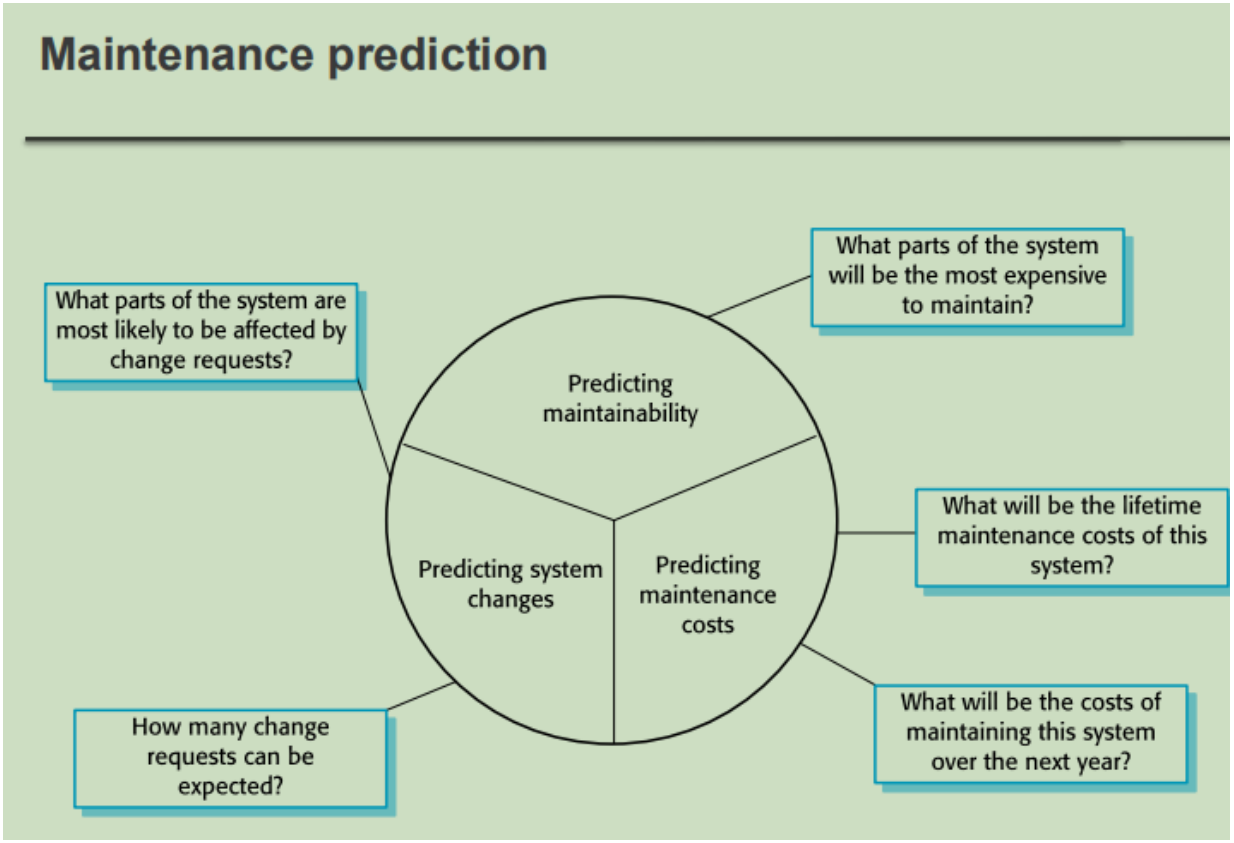
Software maintenance

- Software maintenance: modifying a program after it has been put into use.
- Mostly used for custom software. Generic software: evolve to create new version
- Types

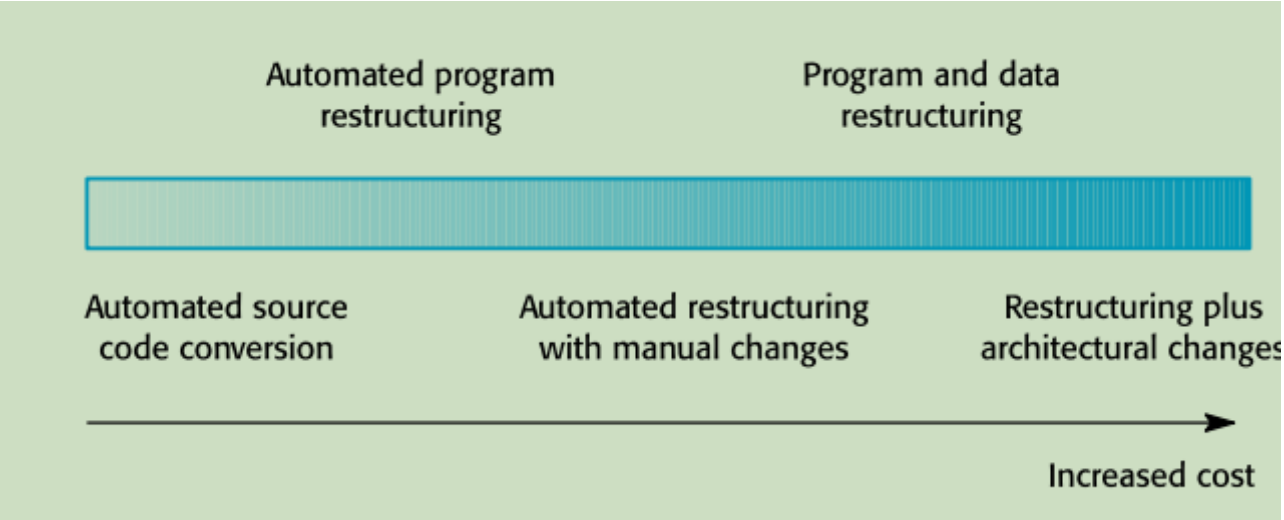
- fault repairs 24%
- environmental adaption: to a different operating environement 19%
- functionality addition and modification 58%

Maintenance prediction

- It is concerned with assessing which parts of the system may cause problems and have high maintenance costs



- Predictions of maintainability can be made by assessing the complexity of system components
- Complexity depends on:
  - complexity of control structures
  - complexity of data structures
  - object, method and module size
- Reengineering process activities
  - source code translation (to new language)
  - reverse engineering (analyse the program to understand it)
  - program structure improvement (restructure automatically for understandability)
  - program modularisation
  - data reengineering



- Refactoring
  - It is the process of making improvements to a program to slow down degradation through chagne
  - continuous process of improvement. Aim to avoid structure and code degradation. re-engineering takes place after system has been maintained for some time.

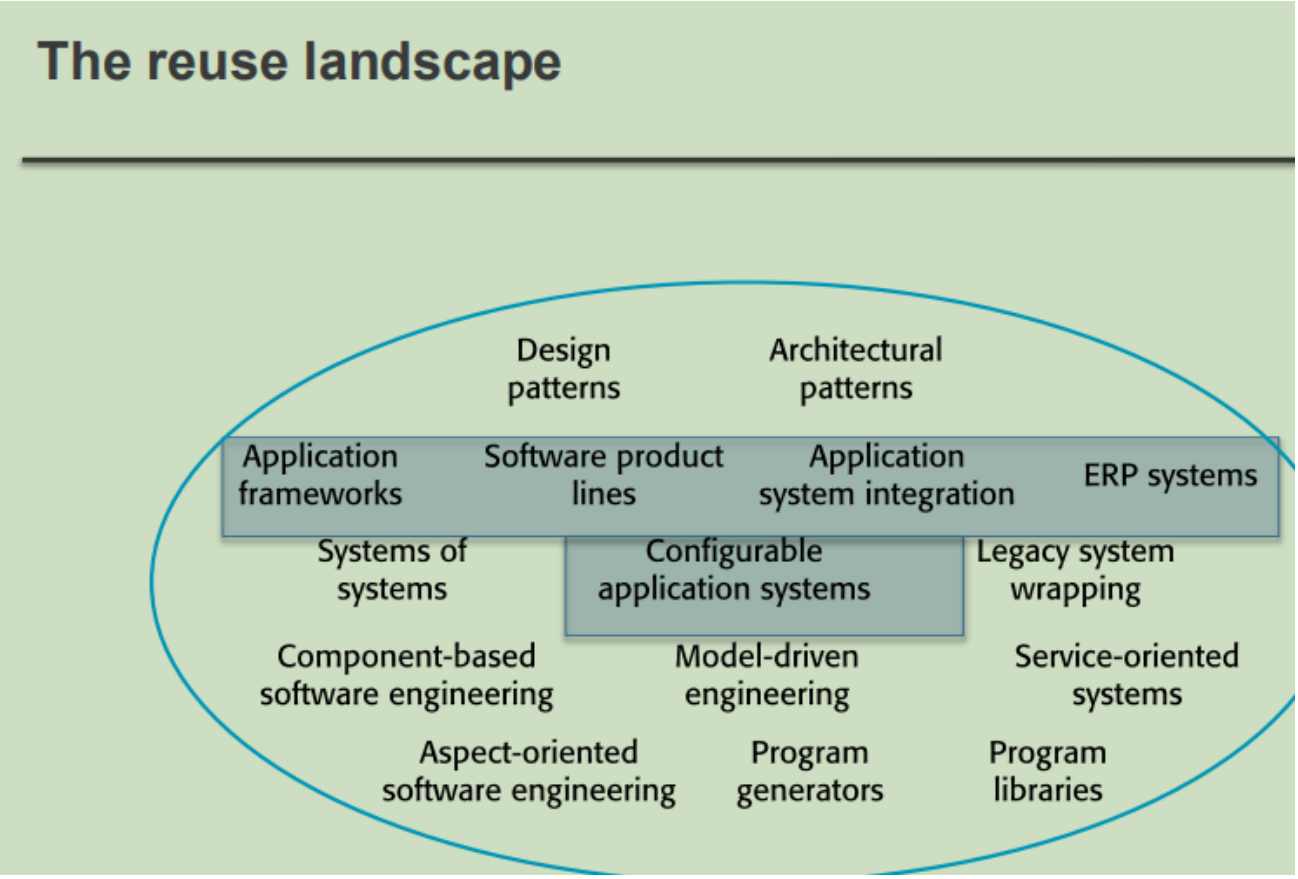
Conclusion

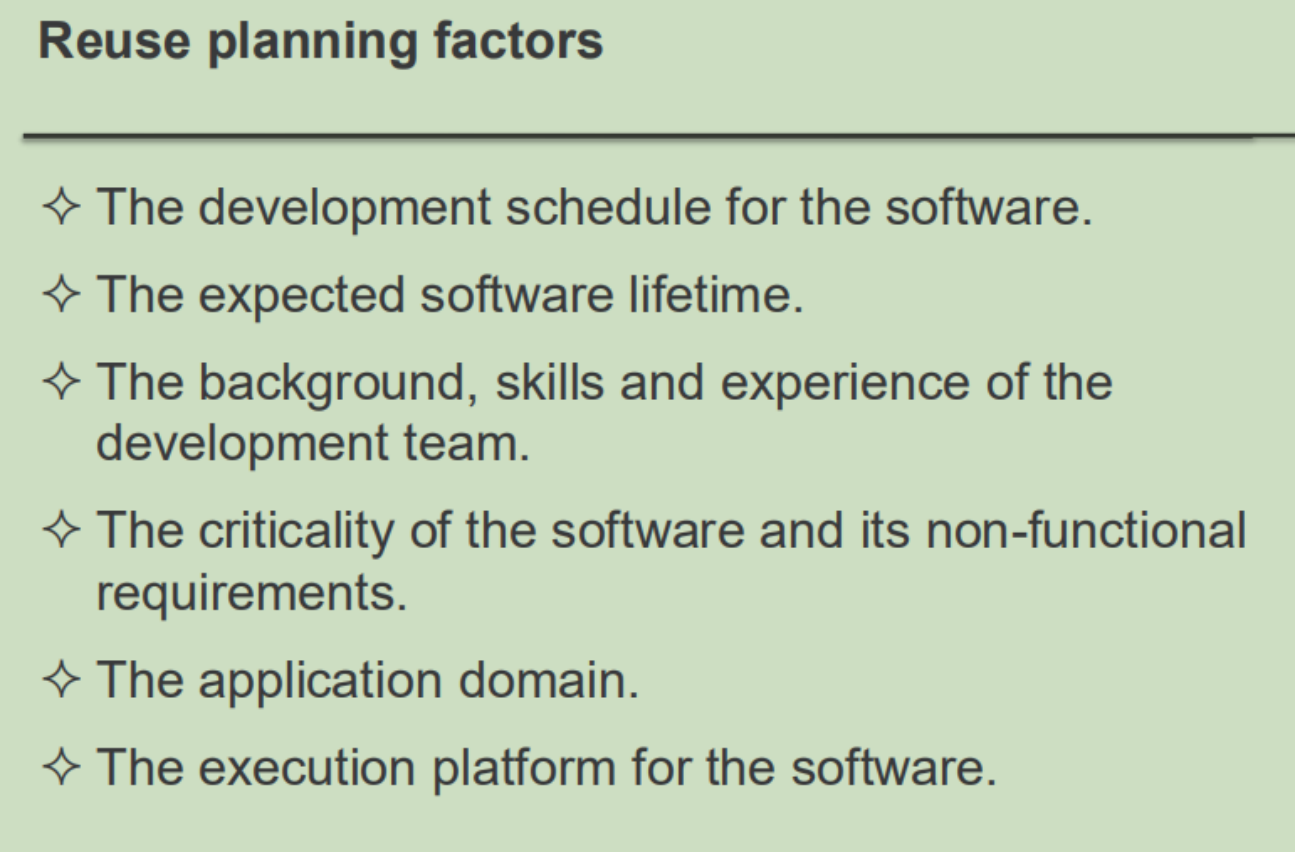
- software development and evolution can be thought of as an integrated, iterative process that can be represented using a spiral model
- for custom systems, the cost of software maintenance usually exceed the software development costs
- the process of software evolution is driven by requests for changes and includes change impact analysis, release planning and change implementation
- legacy system are older software systems, developed using obsolete software and hardware technologies, that remain useful for a business
- it's often cheaper and less risky to maintain a legacy system than to develop a replacement system using modern technology
- the business value of legacy system and the quality of the application should be assessed to help decide if a system should be replaced, transformedor maintained
- 3 types of software maintenance: bug fixing, modifying software to work in a new environment, implementing new or changed requirements
- software re-engineering is concerned with re-structuring and re-documenting software to make it easier to understand and chagne
- refactoring, making program changes that preseve functionality, is a form of preventative maintenance

L11 Software Reuse

- System/application/component/object and fucntion reuse

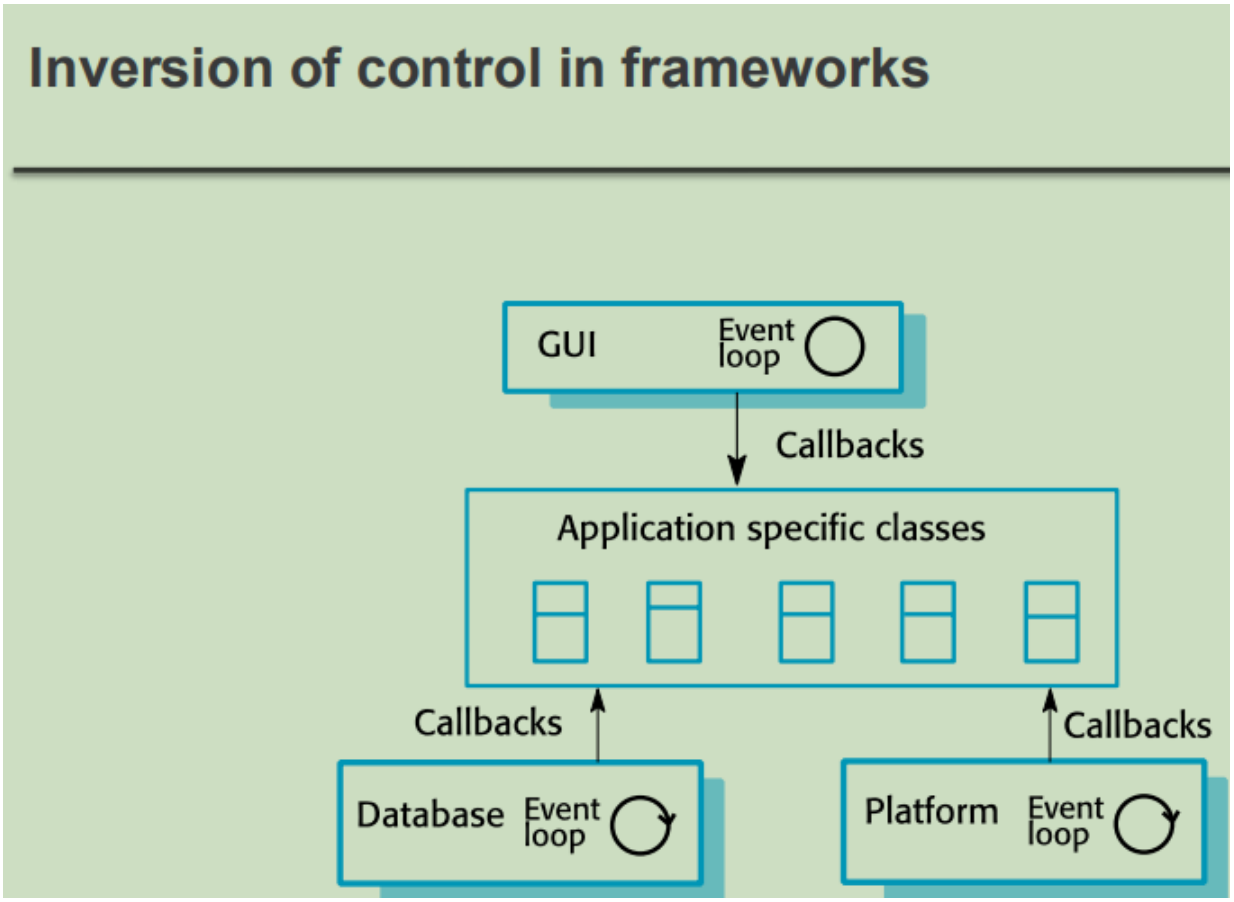
The reuse landscape





Application frame works

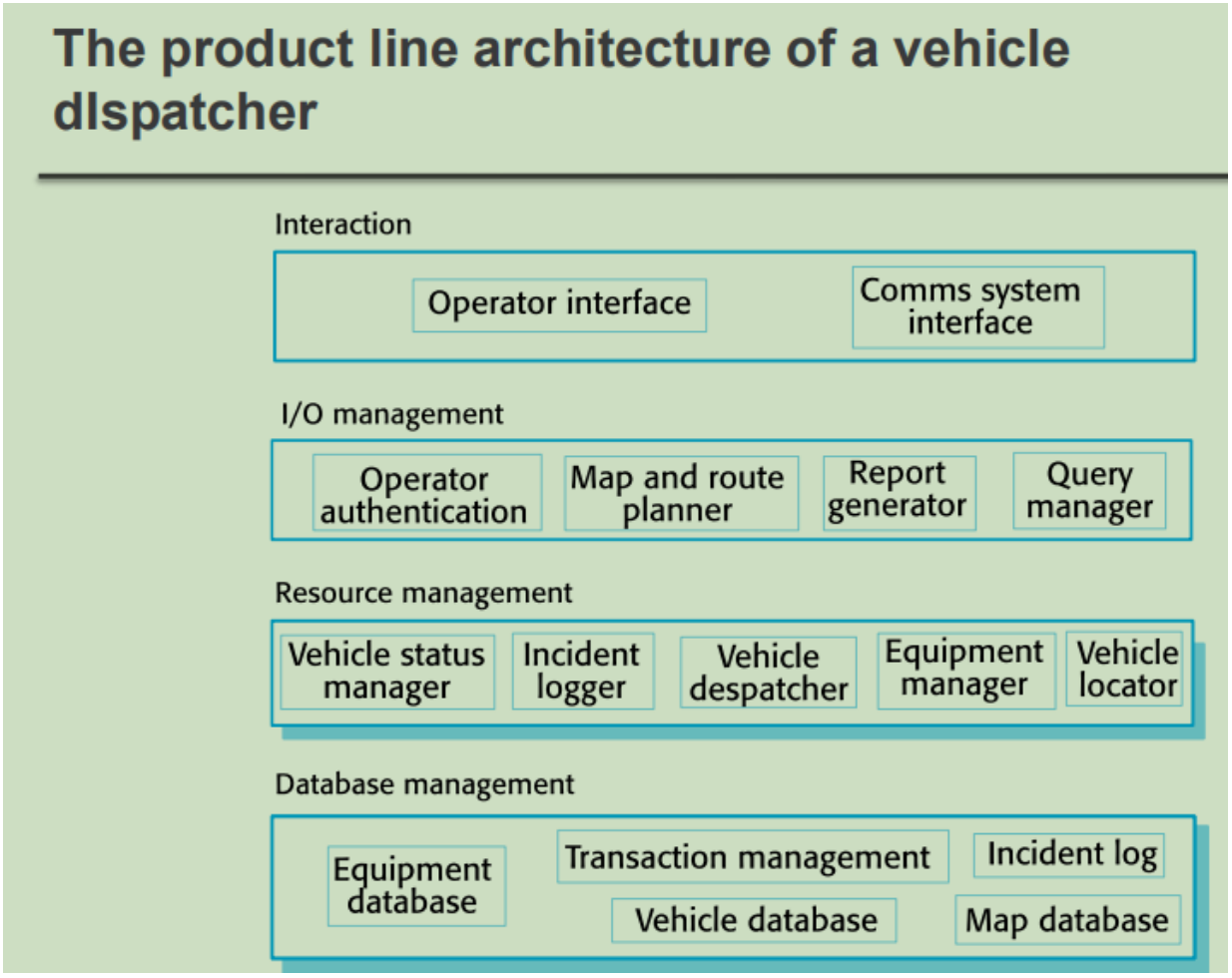
- framework: an integrated set of software artefacts (such as classes, objects) that collaborate to provide a reusable architecutre for a family of related applications. 介乎于system and component
- Extending the framework involves
  - adding concrete classes that inherit operations from abstract classes in the framework
  - adding methods that are called in response to events that are recognised by the framework
- problems with frameworks is their complexity which means that it takes a long time to use them effectively



Software product lines

- software product lines or application families are applications with generic functionality that can be adapted and configured for use in a specific context
- A software product line is a set of applications with a common architecutre and shareed components, with each application specialized to reflect different

requirements



Application system reuse

- An application system product is a software system that can be adapted for different custormers without changing the source code of the system

Conclusion

- There are many different ways to reuse software. These range from the reuse of classes and methods in libraries to the reuse of complete applciation systems
- The advantages of software reuse are lower costs, faster software development and lower risks. System dependability is increased. Specialists can be used more effectively by concentrating their expertise on the design of reusable components
- Application framework are collections of concrete and abstract objects that are designed for reuse through specialization and the addition of new objects. They usually incorporate good design practice through design patterns
- Software product lines are related applications that are developed from one or more base applications. A generic system is adpated and specialized to meet specific requirements for functionality, target platform or operational configuration

- Application system reuse is concerned with the reuse of large-scale, off-the-shell systems. These provide a lot of functionality and their reuse can radically reduce costs and development time. Systems may be developed by configuring a single, generic application system or by integrating two or more application systems
- Potential problems with application system reuse include lack of control over functionality and performance, lack of control over system evolution, the need for support from external vendors and difficulities in ensuring that systems can inter-operate.