



Lecture 7 – Operation, Maintenance and Evolution (Sommerville Ch. 27)

Karl R. Wilcox
K.R.Wilcox@reading.ac.uk



Objectives

- To look at system operation
- To understand some of the issues of system maintenance
- To understand at a high level system evolution and in particular, migration of legacy systems
- Today's seminar
 - Job Titles and Roles in Software Engineering



System Operation

- *Should* be the least troublesome phase in the lifecycle!
- Often the aim is for “lights out” operation
 - In contrast to the heavily staffed machine room of the past
- There are some pitfalls:
 - Don’t forget to do all the things that you put off doing during development
 - Watch growth in general, beware of hidden “step functions”
 - Try to find out about any manual “workarounds” that have to be used
- Plan (and practice) for disaster recovery



Software change

- **Software change is inevitable**
 - New requirements emerge when the software is used
 - The business environment changes
 - Errors must be repaired
 - New equipment must be accommodated
 - The performance or reliability may have to be improved
- **A key problem for organisations is implementing and managing change to their legacy systems**



Software change strategies

- **Software maintenance**
 - Changes are made in response to changed requirements but the fundamental software structure is stable
- **Architectural transformation**
 - The architecture of the system is modified generally from a centralised architecture to a distributed architecture
- **Software re-engineering**
 - No new functionality is added to the system but it is restructured and reorganised to facilitate future changes
- **These strategies may be applied separately or together**



Software maintenance

- **Modifying a program after it has been put into use**
- **Maintenance does not normally involve major changes to the system's architecture**
- **Changes are implemented by modifying existing components and adding new components to the system**



Maintenance is inevitable

- **The system requirements are likely to change while the system is being developed because the environment is changing. Therefore a delivered system won't meet its requirements!**
- **Systems are tightly coupled with their environment. When a system is installed in an environment it changes that environment and therefore changes the system requirements.**
- **Systems MUST be maintained therefore if they are to remain useful in an environment**

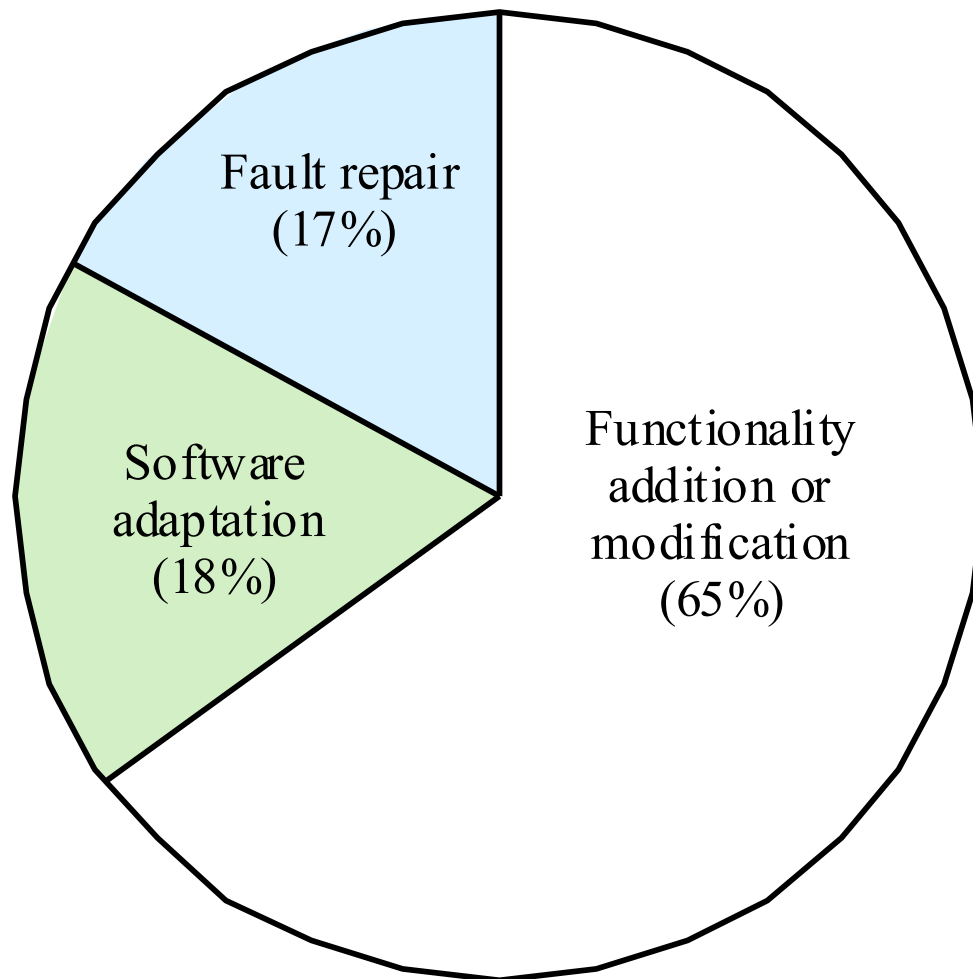


Types of maintenance

- **Maintenance to repair software faults**
 - Changing a system to correct deficiencies in the way meets its requirements
- **Maintenance to adapt software to a different operating environment**
 - Changing a system so that it operates in a different environment (computer, OS, etc.) from its initial implementation
- **Maintenance to add to or modify the system's functionality**
 - Modifying the system to satisfy new requirements

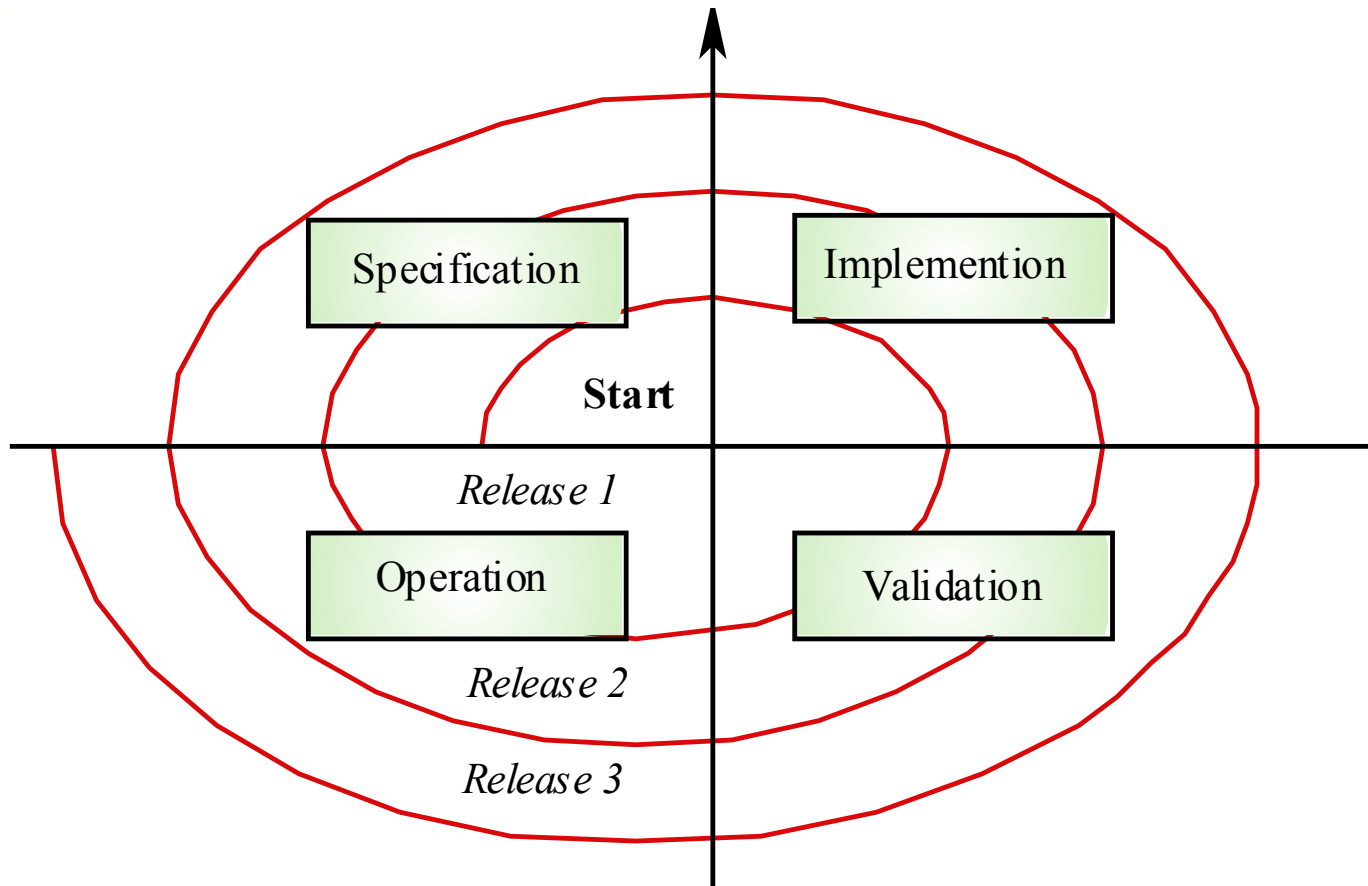


Distribution of maintenance effort





Spiral maintenance model





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.)



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

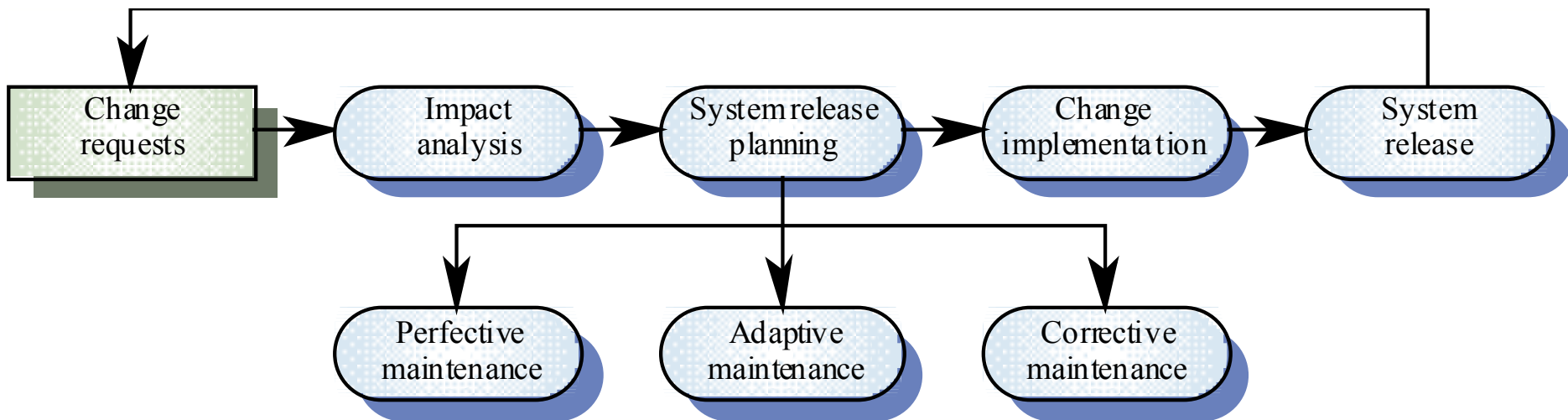


Evolutionary software

- **Rather than think of separate development and maintenance phases, evolutionary software is software that is designed so that it can continuously evolve throughout its lifetime**



The maintenance process





Change requests

- **Change requests are requests for system changes from users, customers or management**
- **In principle, all change requests should be carefully analysed as part of the maintenance process and then implemented**
- **In practice, some change requests must be implemented urgently**
 - **Fault repair**
 - **Changes to the system's environment**
 - **Urgently required business changes**



Architectural evolution

- **There is a need to convert many legacy systems from a centralised architecture to a client-server architecture**
- **Change drivers**
 - **Hardware costs. Servers are cheaper than mainframes**
 - **User interface expectations. Users expect graphical user interfaces**
 - **Distributed access to systems. Users wish to access the system from different, geographically separated, computers**



Distribution factors

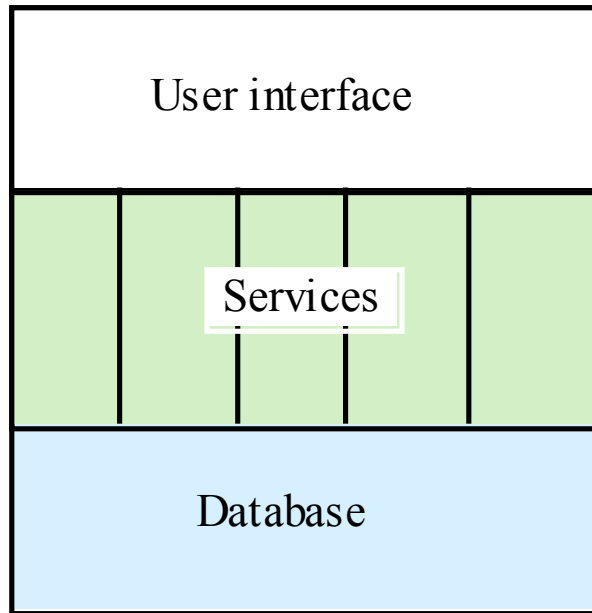
Factor	Description
Business importance	Returns on the investment of distributing a legacy system depend on its importance to the business and how long it will remain important. If distribution provides more efficient support for stable business processes then it is more likely to be a cost-effective evolution strategy.
System age	The older the system the more difficult it will be to modify its architecture because previous changes will have degraded the structure of the system.
System structure	The more modular the system, the easier it will be to change the architecture. If the application logic, the data management and the user interface of the system are closely intertwined, it will be difficult to separate functions for migration.
Hardware procurement policies	Application distribution may be necessary if there is company policy to replace expensive mainframe computers with cheaper servers. .



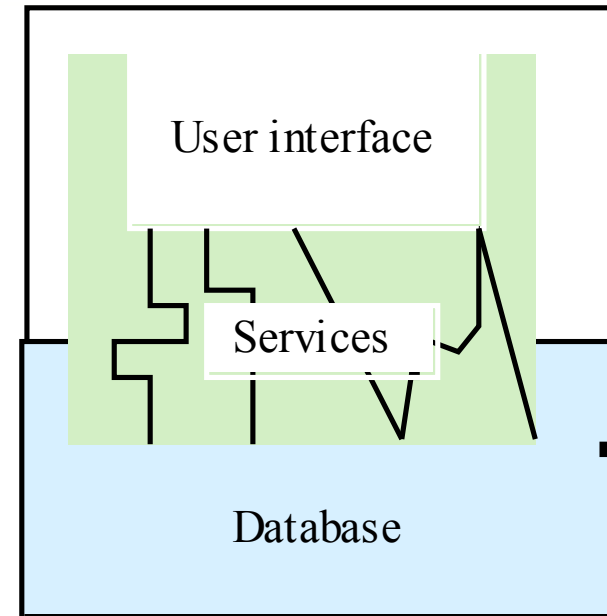
Legacy system structure

- Ideally, for distribution, there should be a clear separation between the user interface, the system services and the system data management
- In practice, these are usually intermingled in older legacy systems

Legacy system structures

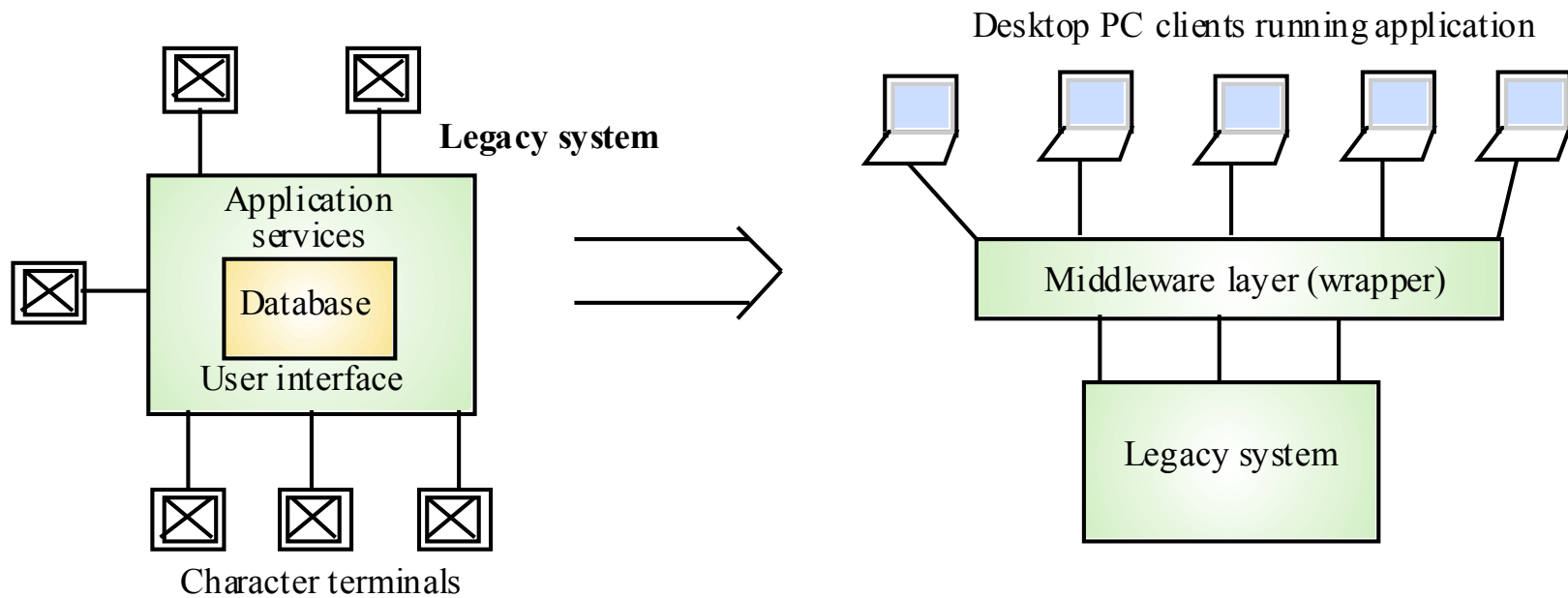


Ideal model for distribution



Real legacy systems

Legacy system distribution





Distribution options

- The more that is distributed from the server to the client, the higher the costs of architectural evolution
- The simplest distribution model is UI distribution where only the user interface is implemented on the server
- The most complex option is where the server simply provides data management and application services are implemented on the client



Distribution option spectrum

Server: Interaction control
Data validation
Services
Database

Server: Services
Database

Server: Database



Client: Presentation

Client: Presentation
Interaction control
Data validation

Client: Presentation
Interaction control
Data validation
Services



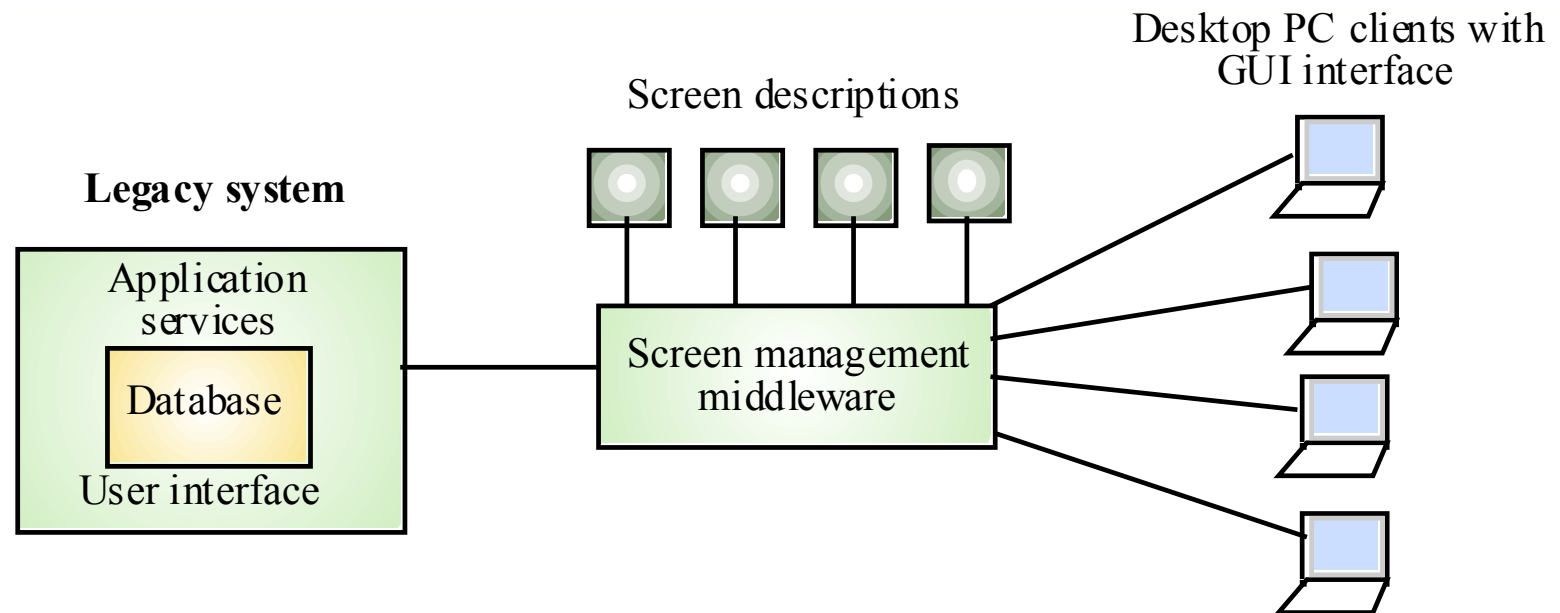
Increasing cost
and effort



User interface distribution

- UI distribution takes advantage of the local processing power on PCs to implement a graphical user interface
- Where there is a clear separation between the UI and the application then the legacy system can be modified to distribute the UI
- Otherwise, screen management middleware can translate text interfaces to graphical interfaces

User interface distribution





UI migration strategies

Strategy	Advantages	Disadvantages
Implementation using the window management system	Access to all UI functions so no real restrictions on UI design Better UI performance	Platform dependent May be more difficult to achieve interface consistency
Implementation using a web browser	Platform independent Lower training costs due to user familiarity with the WWW Easier to achieve interface consistency	Potentially poorer UI performance Interface design is constrained by the facilities provided by web browsers



Key points

- The costs of software change usually exceed the costs of software development
- Factors influencing maintenance costs include staff stability, the nature of the development contract, skill shortages and degraded system structure
- Architectural evolution is concerned with evolving centralised to distributed architectures
- A distributed user interface can be supported using screen management middleware