Lecture XII:

Software Evolution

General Views

Software developers
like to solve problems.

If there are
no problems handily available

they will create **their own problems**!

Topics Covered

- Evolution processes
  - Change processes for software systems
- Software maintenance
  - Making changes to operational software systems
- Legacy system management
  - Making decisions about software change

Software Change

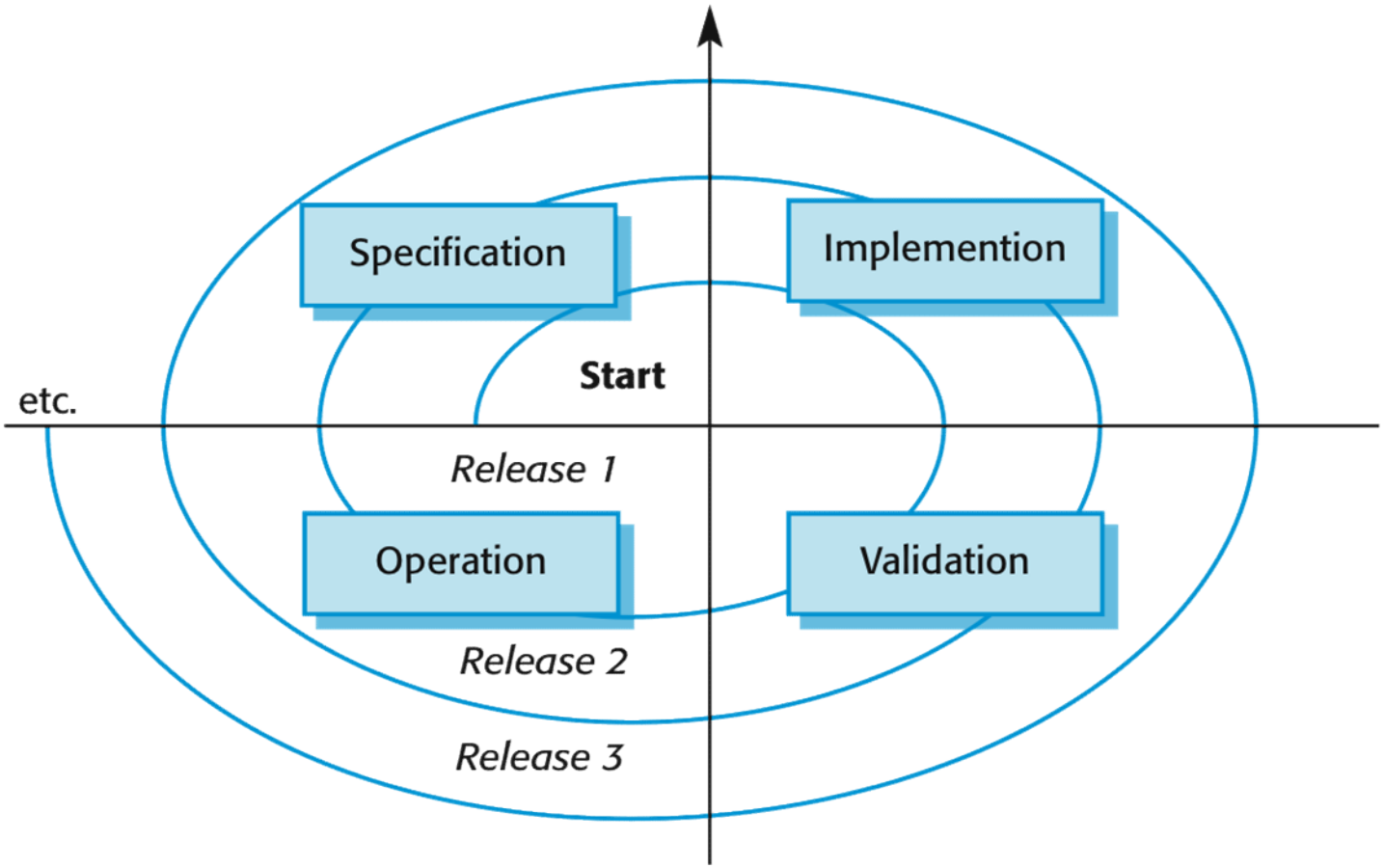
Software Change

- Software change is inevitable:
  - New requirements emerge when the software is used
  - The business environment changes
  - Errors must be repaired
  - New computers and equipment are added to the system
  - The performance or reliability of the system may have to be improved
- A key problem for all organizations is implementing and managing change to their existing software systems

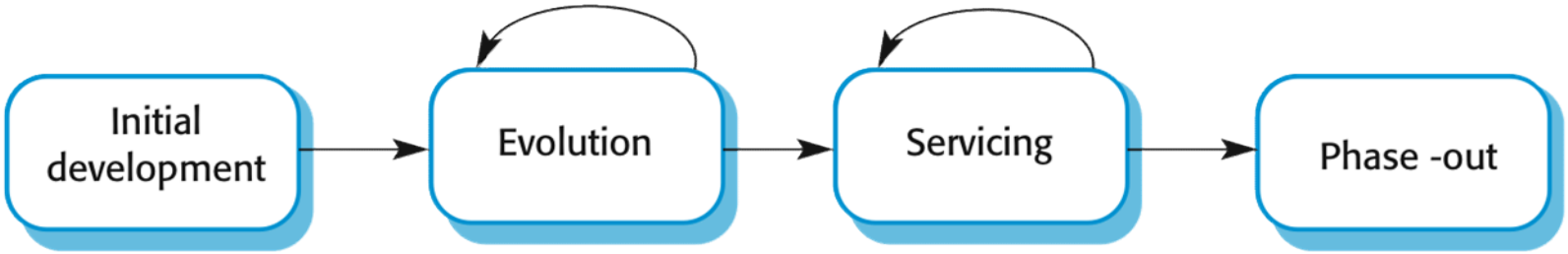
Importance of Evolution

- Organizations have huge investments in their software systems - they are critical business assets
- To maintain the **value** of these assets to the business, they must be changed and updated
- The majority of the software budget in large companies is devoted to **changing and evolving existing software** rather than developing new software

A Spiral Model of Development and Evolution



Evolution and Servicing



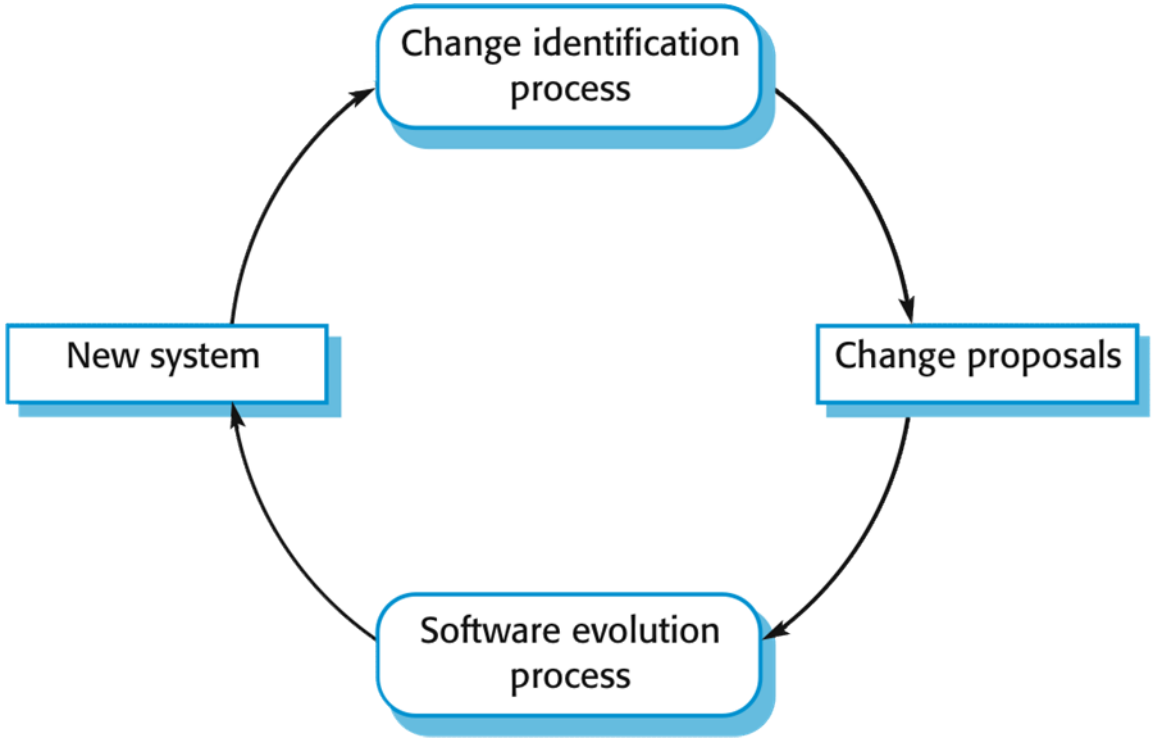
Evolution and Servicing

- Evolution
  - The stage in a software system's life cycle where it is in operational use and is evolving as new requirements are proposed and implemented in the system
- Servicing
  - At this stage, the software remains useful but the only changes made are those required to keep it operational, i.e. bug fixes and changes to reflect changes in the software's environment. No new functionality is added
- Phase-out
  - The software may still be used but no further changes are made to it

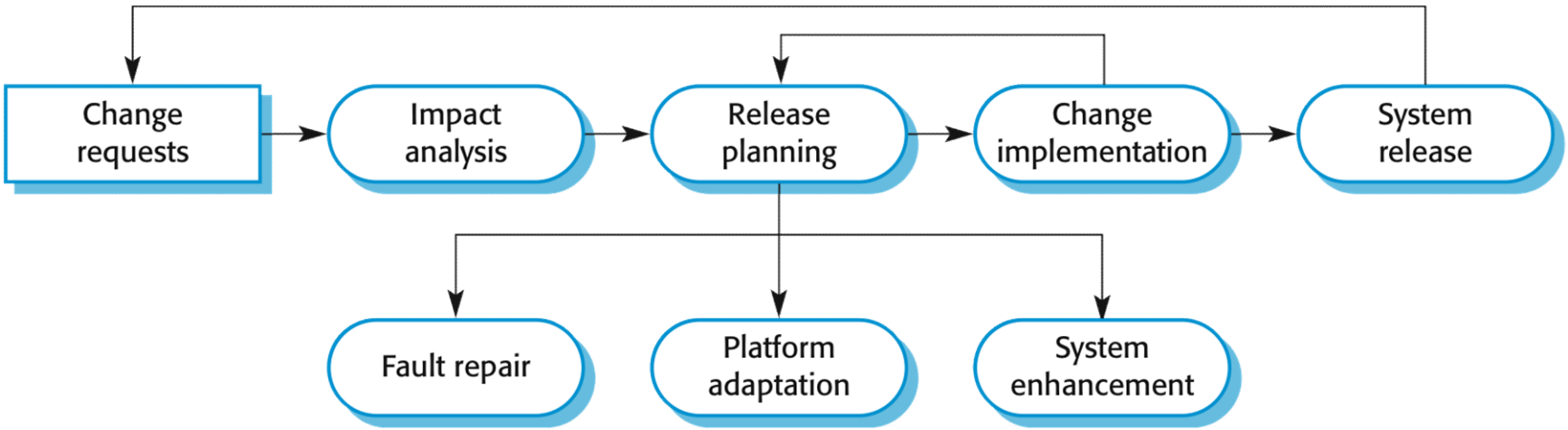
Evolution Processes

- Software evolution processes depend on
  - The type of software being maintained
  - The development processes used
  - The skills and experience of the people involved
- Proposals for change are the driver for system evolution
  - Should be linked with components that are affected by the change, thus allowing the cost and impact of the change to be estimated
- Change identification and evolution continues throughout the system lifetime

Change Identification and Evolution Processes

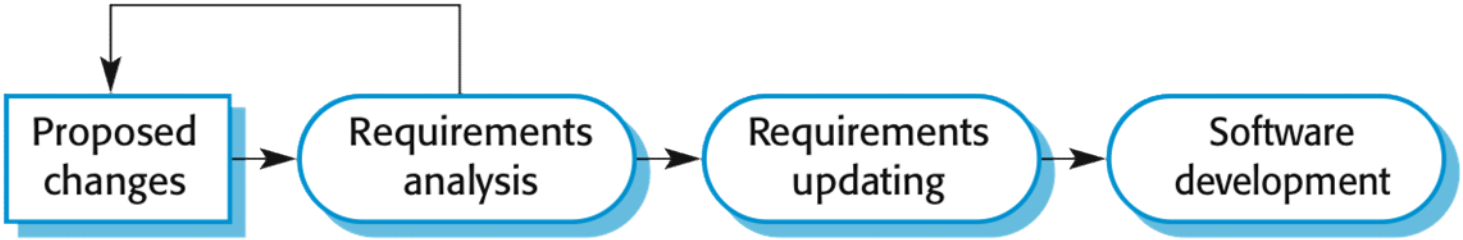


The Software Evolution Process



Change Implementation

Change Implementation



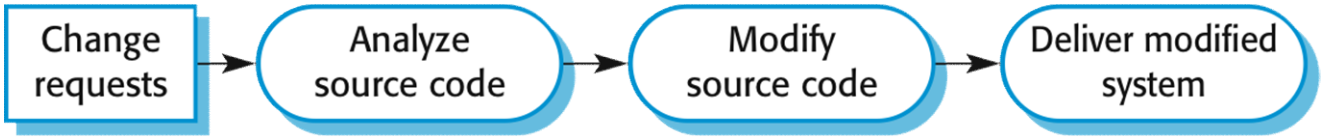
Change Implementation

- Iteration of the development process where the revisions to the system are designed, implemented and tested
- A critical difference is that the first stage of change implementation may involve program understanding, especially if the original system developers are not responsible for the change implementation
- During the program understanding phase, you have to understand how the program is structured, how it delivers functionality and how the proposed change might affect the program

Urgent Change Requests

- Urgent changes may have to be implemented without going through all stages of the software engineering process
  - If a serious system fault has to be repaired to allow normal operation to continue
  - If changes to the system's environment (e.g., an OS upgrade) have unexpected effects
  - If there are business changes that require a very rapid response (e.g. the release of a competing product)

The Emergency Repair Process



Software Maintenance

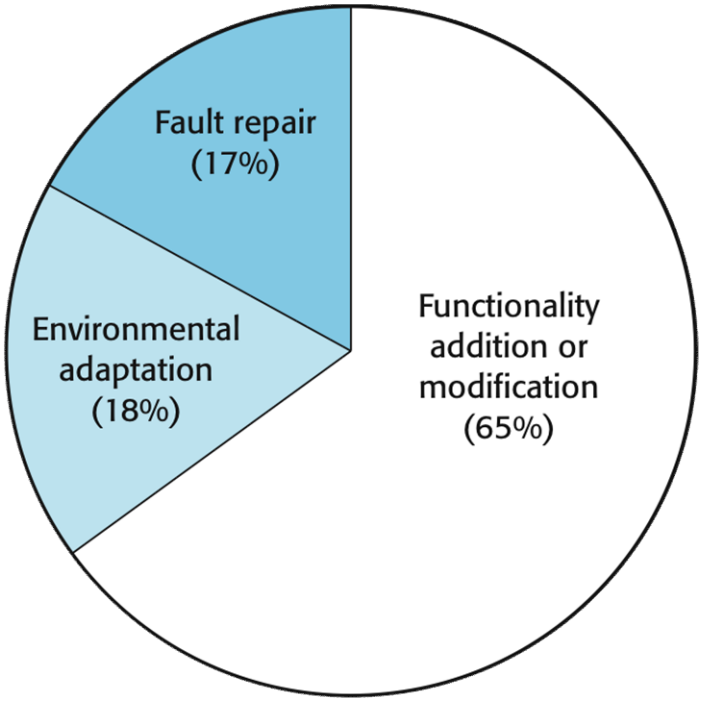
Software Maintenance

- **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 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
- 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

Maintenance Effort Distribution

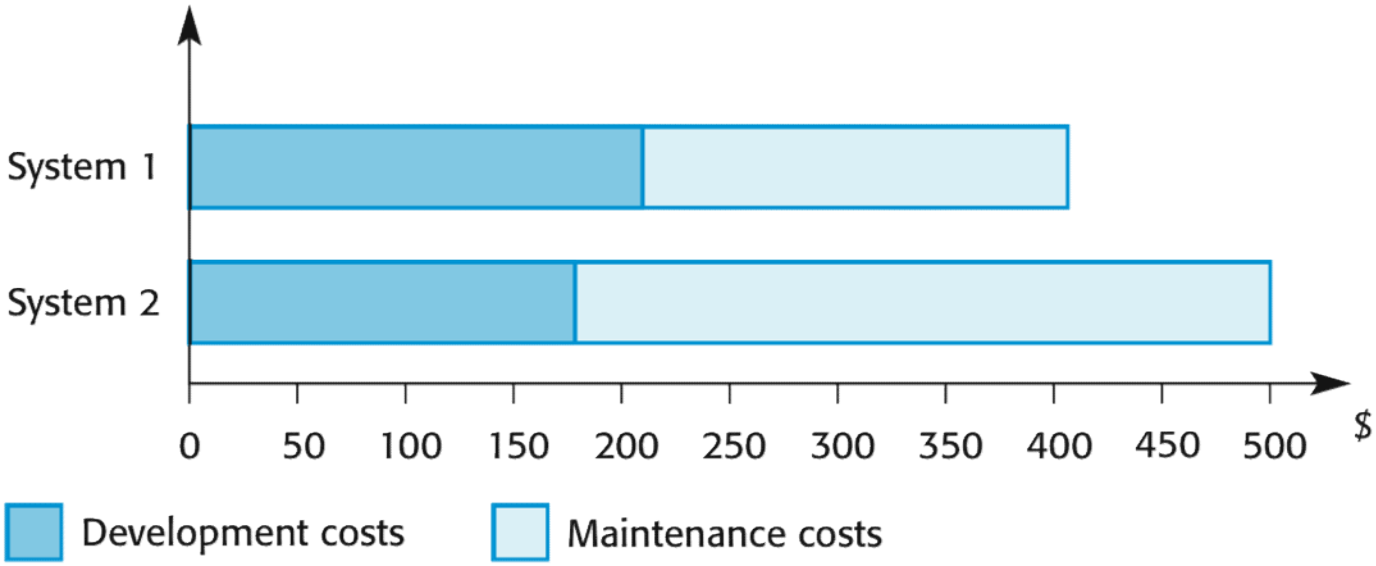


Maintenance Costs

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.).

Development and Maintenance Costs



Maintenance Costs 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

- 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