Software Evolution and Maintenance

Mojtaba Shahin

Week #11: Lecture - Part 1

Topics

- Part 1
  - Software Evolution
  - Legacy Systems

Notes and Acknowledgements

- Slides/images come from the following main sources:
  - **Chapter 9**: Ian Sommerville, Software Engineering, 10th Edition, 2015.
    - https://iansommerville.com/software-engineering-book/slides/

Software change

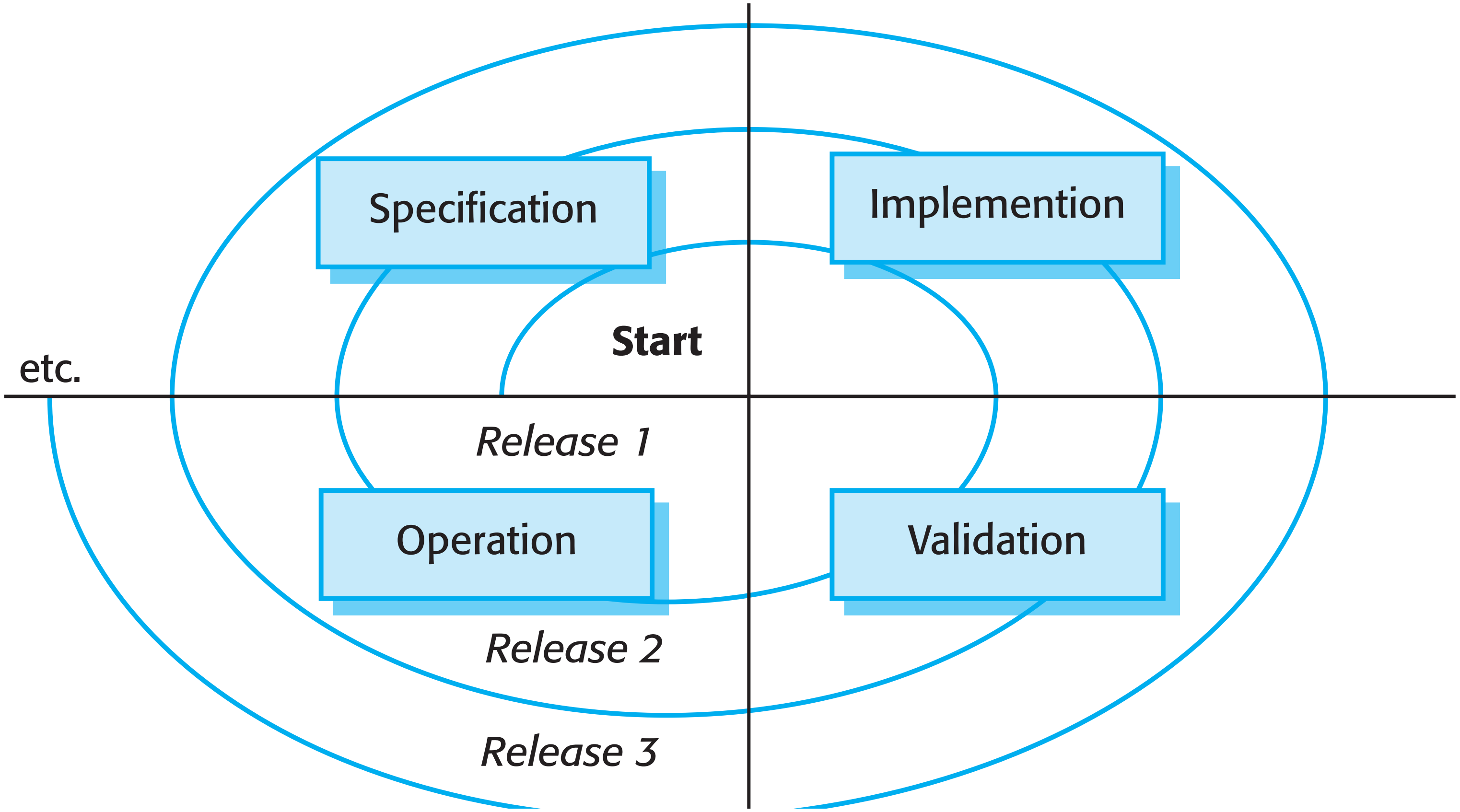
- Software change is **inevitable**
  - New requirements emerge when the software is used;
  - The business environment changes;
  - Errors must be repaired;
  - New computers and equipment are added to the system;
  - The performance or reliability of the system may have to be improved.
- A key problem for all organizations is how to implement and manage **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.

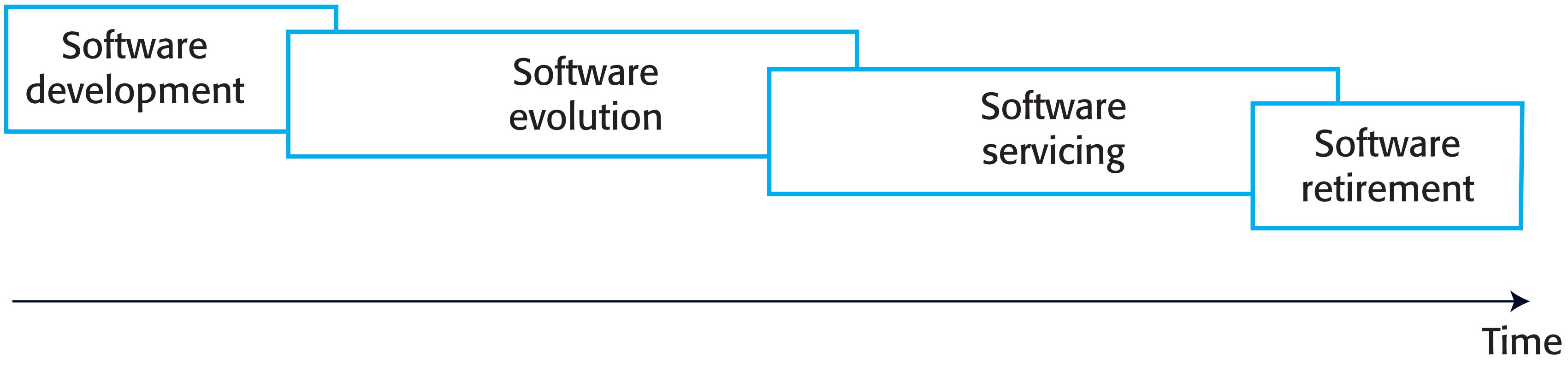
A spiral model of development and evolution

- We start by creating **Release 1** of the system. Once delivered, changes are proposed, and the development of **Release 2** starts almost immediately.



Evolution and Servicing
Alternative view of the software evolution life cycle

- Proposed by Rajlich and Bennett (Rajlich and Bennett 2000)



Evolution and servicing

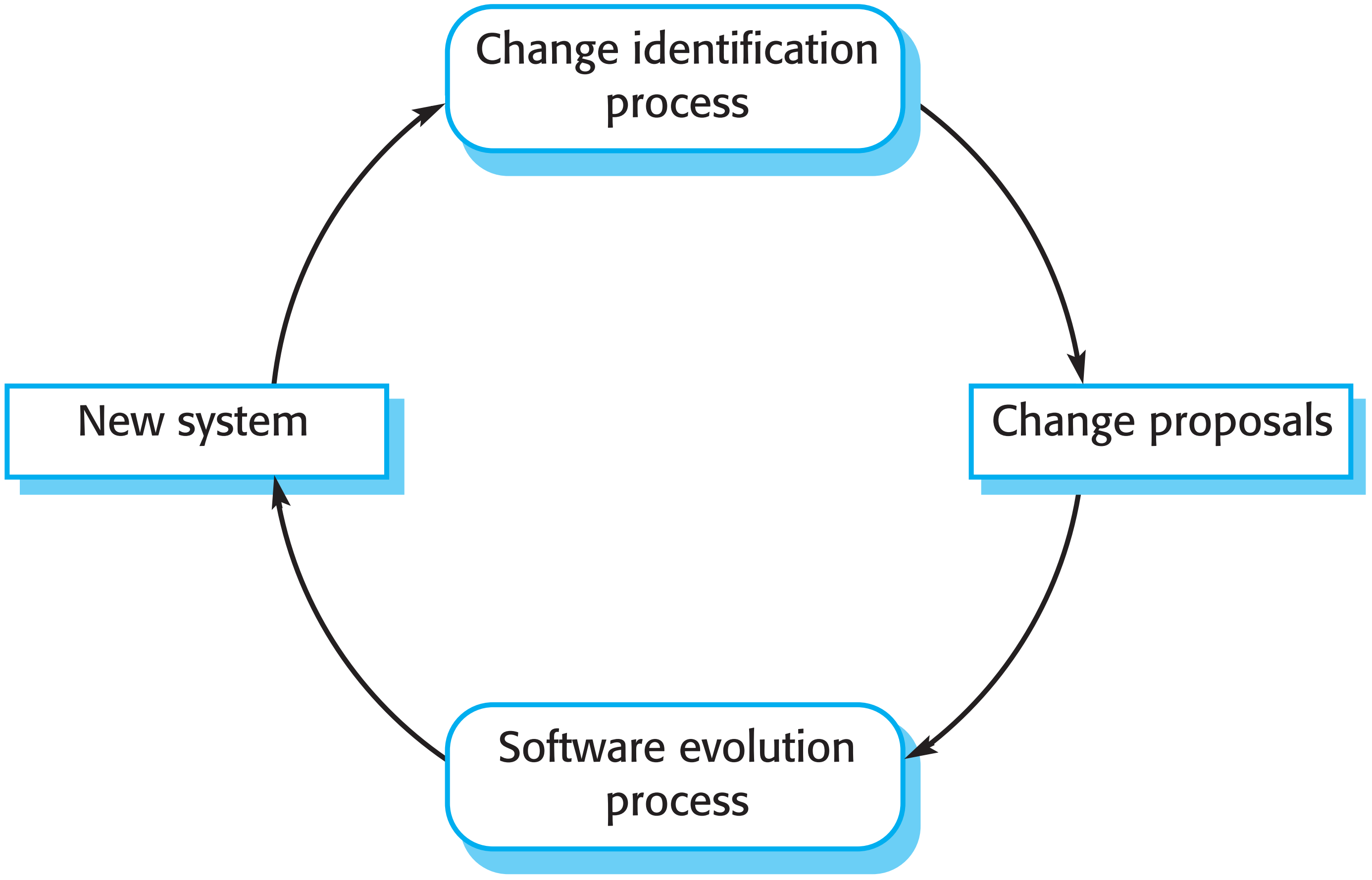
- **Evolution**
  - The stage in a software system's life cycle where it is in operational use and is evolving as new requirements are proposed and implemented in the system.
- **Servicing**
  - At this stage, the software remains useful but the only changes made are those required to keep it operational 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

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** of 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.

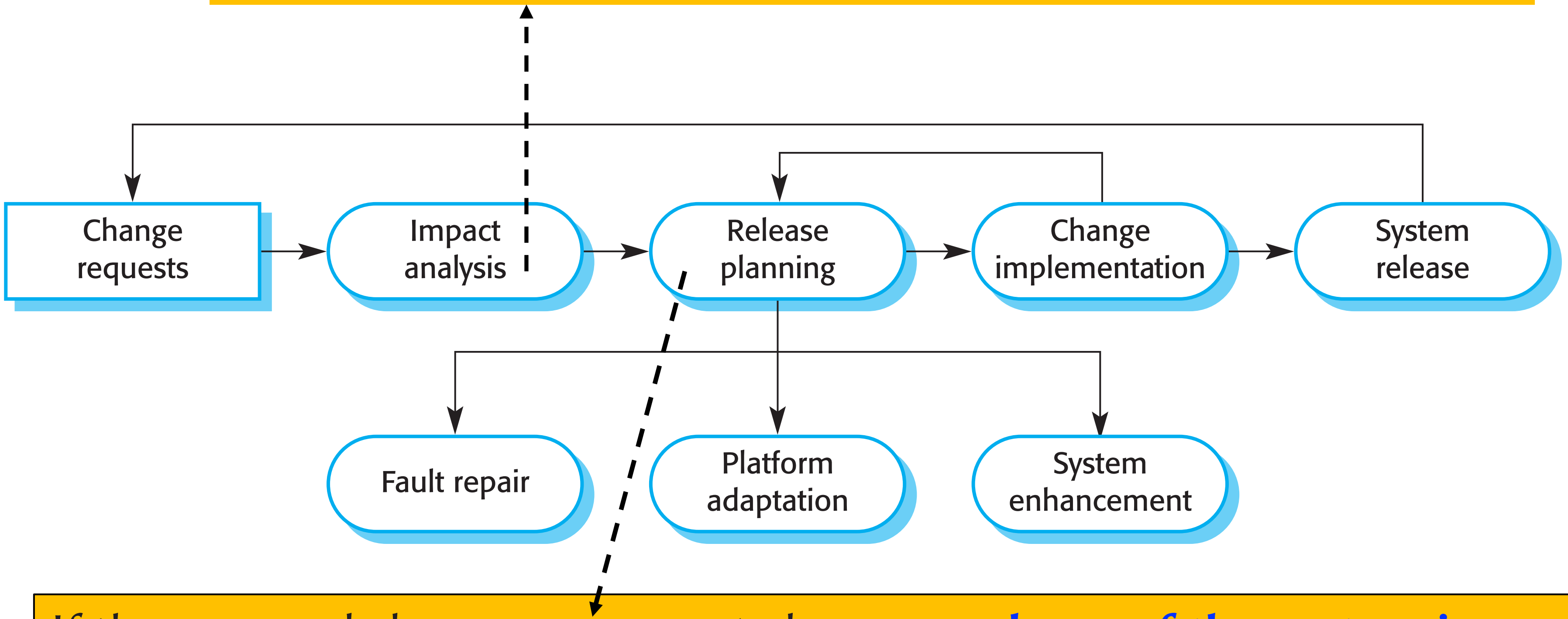
Change identification and evolution processes



Change Proposal

- Change proposals might be
  - based on **existing requirements** that have not been implemented in the released system,
  - requests for **new requirements**,
  - **bug reports** from system stakeholders, and
  - **new ideas for software improvement** from the system development team.

The software evolution process

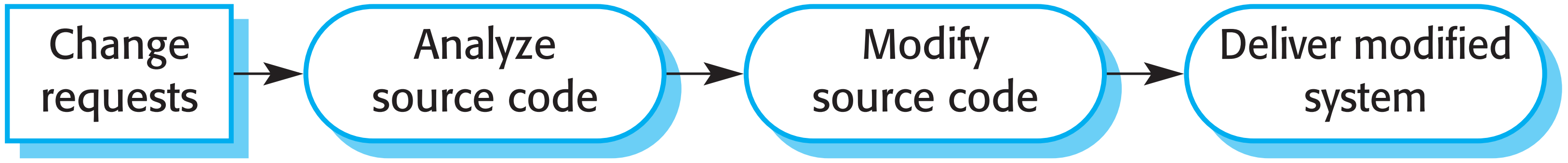


If the proposed changes are accepted, **a new release of the system is planned**. During release planning, all proposed changes (fault repair, adaptation, and new functionality) are considered. A **decision** is then made on which changes to implement in the next version of the system.

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 **very rapid response** (e.g. the release of a competing product).

The emergency repair process

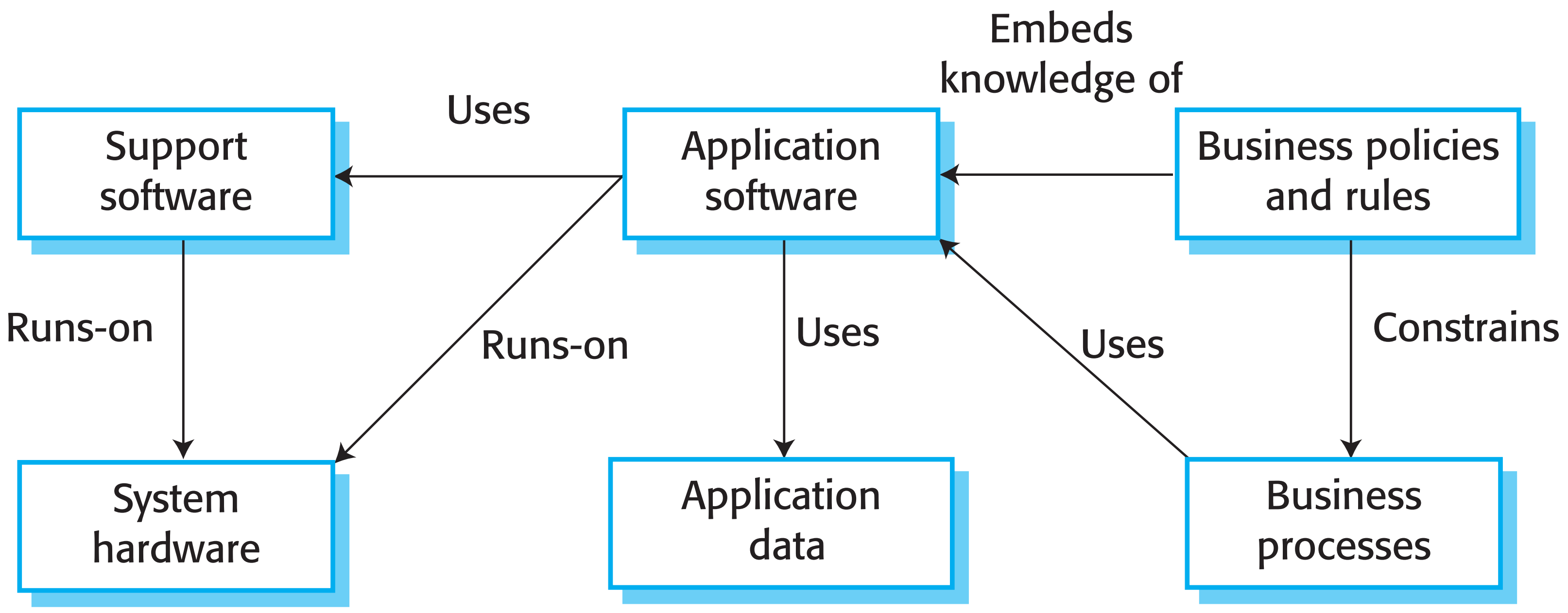


Legacy Systems

Legacy systems

- Legacy systems are older systems that rely on languages and technology that are no longer used for new systems development.
- Legacy software may be dependent on older hardware, such as mainframe computers and may have associated legacy processes and procedures.
- Legacy systems are not just software systems but are broader **socio-technical systems** that include hardware, software, libraries and other supporting software and business processes.

The elements of a legacy system



Legacy system components

- ***System hardware*** Legacy systems may have been written for hardware that is no longer available.
- ***Support software*** The legacy system may rely on a range of support software, which may be obsolete or unsupported.
- ***Application software*** The application system that provides the business 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.

Legacy system components

- ***Business processes*** These are processes that are used in the business to achieve some business objective.
  - Business processes may be designed around a legacy system and constrained by the functionality that it provides.
- ***Business policies and rules*** These are definitions of how the business should be carried out and constraints on the business. Use of the legacy application system may be embedded in these policies and rules.

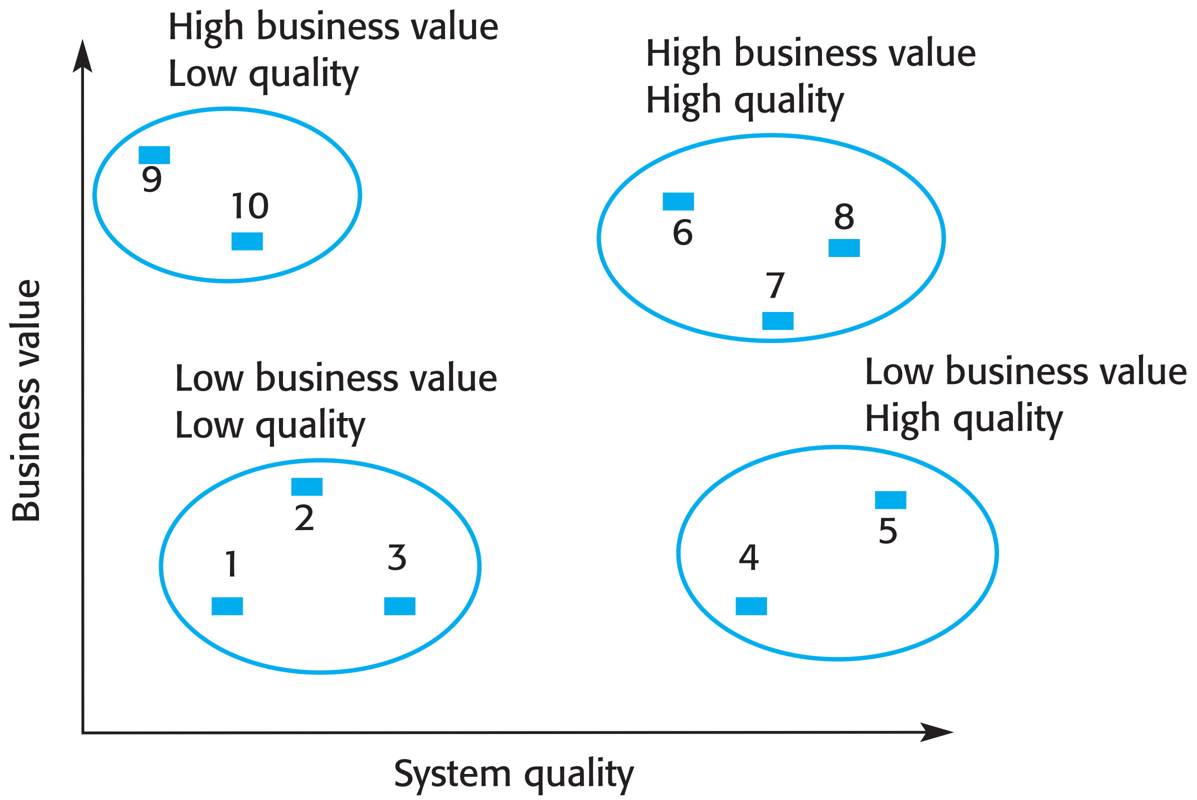
Legacy system change and/or replacement

- Legacy system change and/or replacement is risky and expensive for a number of reasons
  - –Lack of complete system specification and documentation
  - –No consistent programming style
  - –Use of obsolete programming languages with few people available with these language skills
  - –System structure degradation
  - –Data errors, duplication and inconsistency

Legacy system management

- Organisations that rely on legacy systems must choose a strategy for evolving these systems
  1. Scrap the system completely;
  2. Leave the system unchanged and continue with regular maintenance;
  3. Reengineer the system to improve its maintainability;
  4. Replace the system with a new system.
- The strategy chosen should depend on the **system quality** and its **business value**.

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.

References

- Chapter 9: Ian Sommerville, Software Engineering, 10th Edition, 2015.
  - https://iansommerville.com/software-engineering-book/slides/

Thanks!

Mojtaba Shahin

mojtaba.shahin@rmit.edu.au