COMP41670 Software Engineering

17. Software Evolution

Dr Avishek Nag

UCD School of Computer Science. **Scoil na Ríomheolaíochta UCD.**

Table of Contents

Introduction

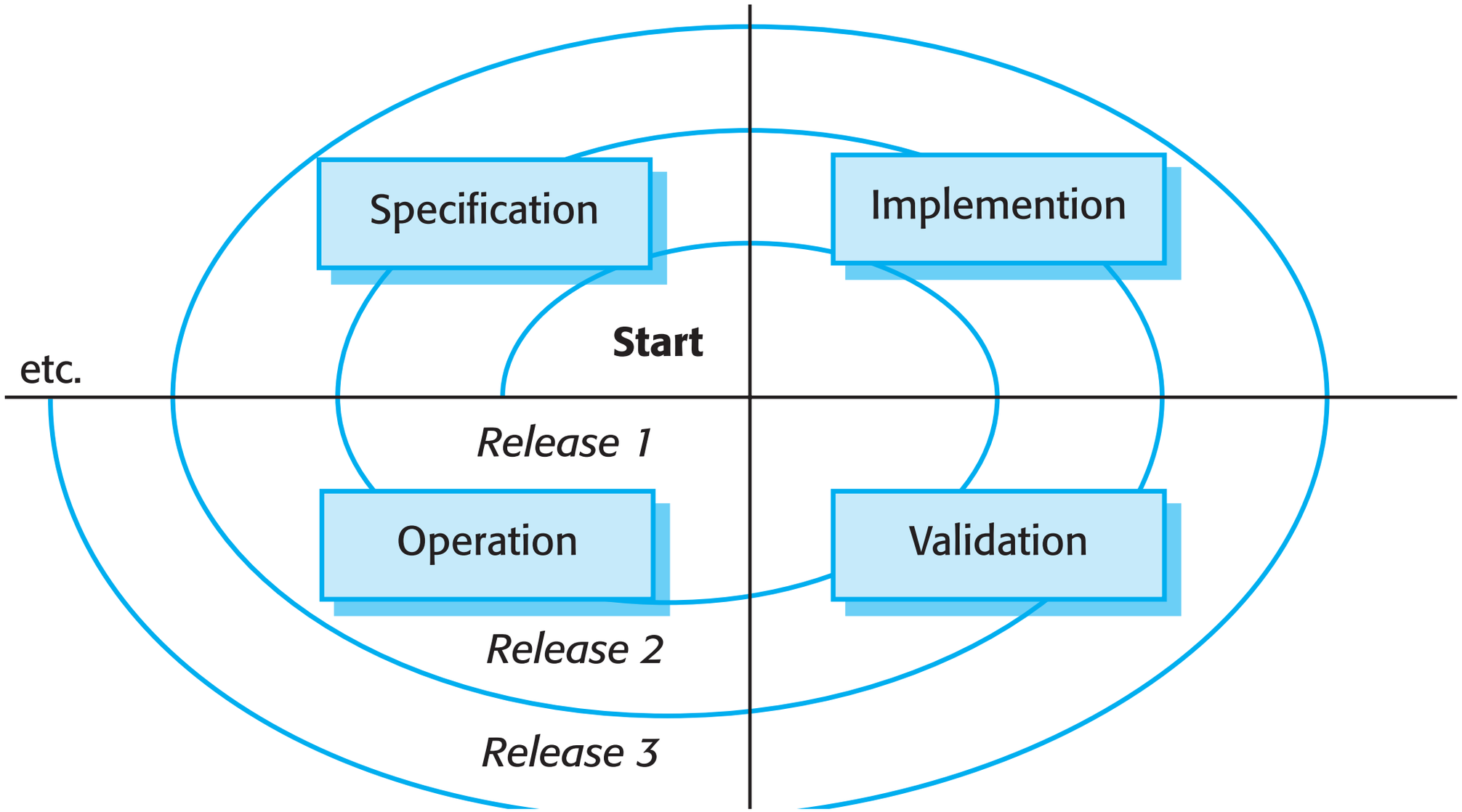
Software Change

- Software change is inevitable due to:
  - New requirements emerging when the software is used
  - The business environment changing
  - The regulatory environment changing, e.g. GDPR, Cookies
  - Bug being found
  - Migration to new versions of the software platform (e.g. new OS)
  - Migration to new hardware
  - The performance or reliability of the system may have to be improved, perhaps due to increased load
- A key problem for all organisations is implementing and managing change to their existing software systems.

Importance of Evolution

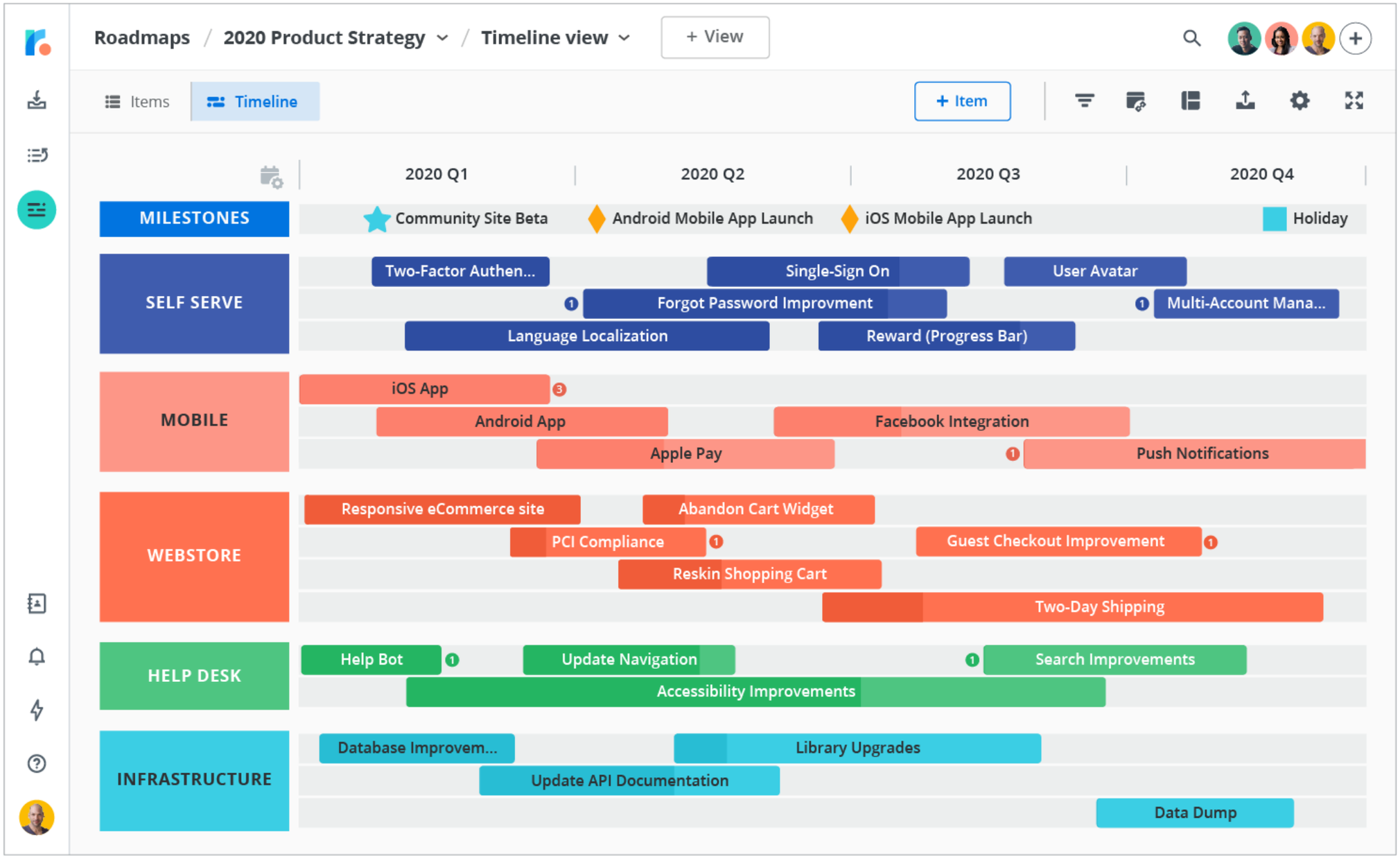
- Organisations have huge investments in their software systems - they are critical business assets.
- To maintain the value of these assets to the business, they must be changed and updated.
- The majority of the software budget in large companies is devoted to changing and evolving existing software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

Spiral Model of Development and Evolution



Product Roadmap

- The product roadmap shows graphically the schedule for product and feature releases to the customer.



[https://roadmunk.com/guides/what-is-a-product-roadmap/]

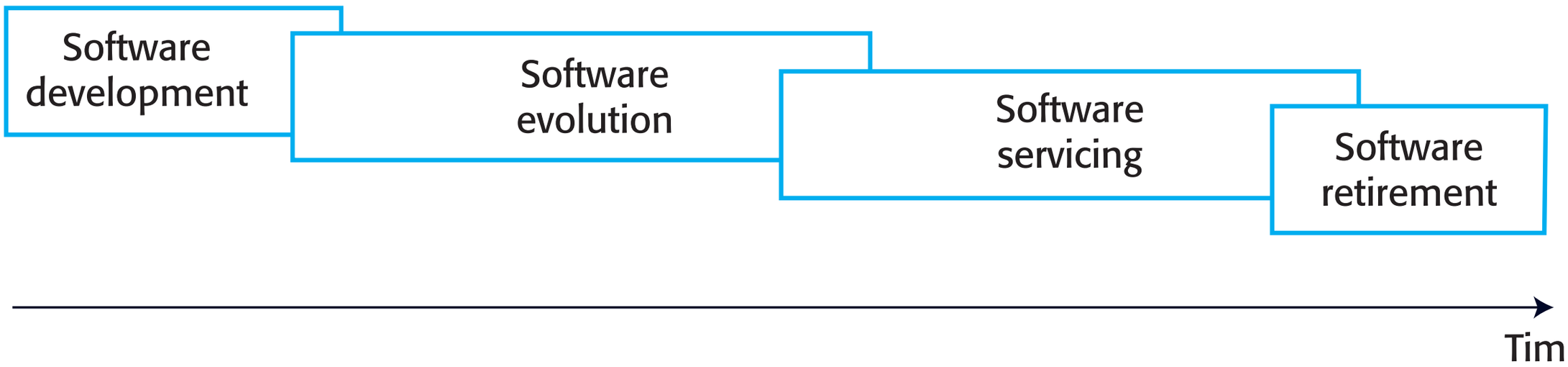
Product Roadmap

- Release numbering:
  - <major release number>.<minor release number>
  - E.g. version 1.1, 1.2, 1.3, 2.1, 2.2, etc
  - Usually, the major release contains significant new features.
  - The minor release contains bug fixes and some addition features.

Product Roadmap

- Pre-alpha releases: Development releases (e.g. at the end of sprints). Usually only for internal testing by the development team and stakeholder review.
- Alpha release: The software is functional but likely to contain bugs. Usually for internal testing beyond the development team.
- Beta release: The software is functional and complete but likely to contain a small number of bugs. Usually for testing by key external users.
- Release candidate: Testing nearly complete.
- Production or live release: Testing complete. General release to customers.

Software Lifecycle



Software Lifecycle

- 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 / Retirement: The software may still be used but no further changes are made to it.

Software Lifecycle

- Deprecated refers to a software or programming language feature that is tolerated or supported but not recommended. A deprecated attribute or feature is one that may eventually be phased out, but continues to be used in the meantime. Deprecation also helps to ward off backward compatibility issues, giving users time to migrate and begin using the newer recommended feature. The deprecated feature will continue to work in the current environment, but will show a warning message that the feature being used may be removed in future releases.

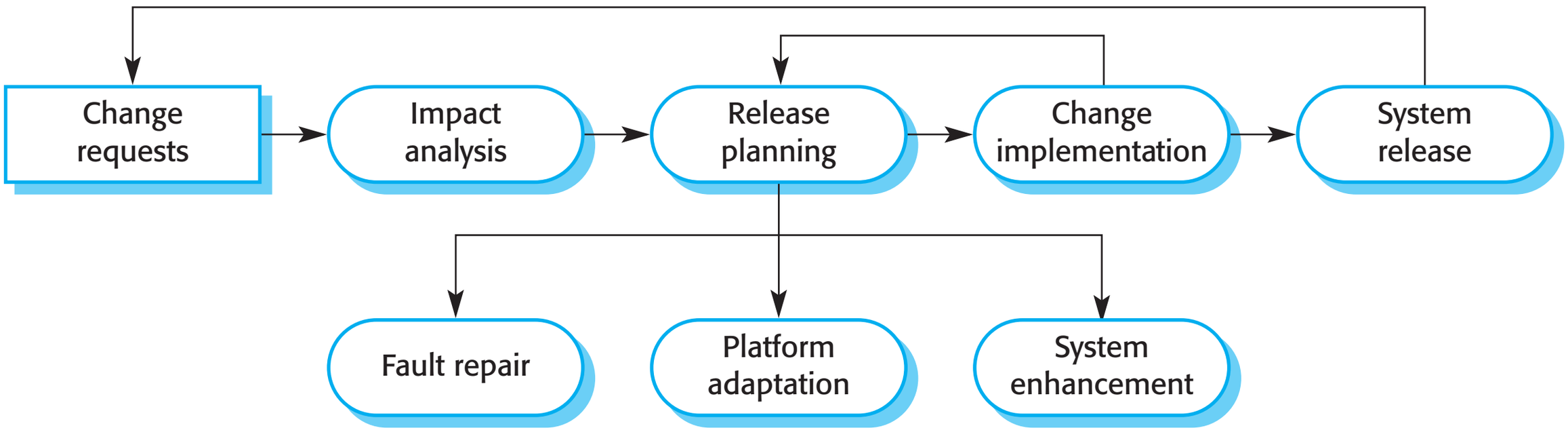
[https://www.techopedia.com/]

Evolution Processes

Change Proposals

- Proposals for change are the driver for system evolution.
- Each change proposal should be reviewed to assess the following before going ahead:
  - The priority of the change
  - The components that are affected by the change
  - An estimate effort required including design, development, testing, documentation and roll out
  - An estimate of the delivery date
  - An estimate of the risk involved in the change (what could go wrong?)
- A go / no go decision is made after the review.

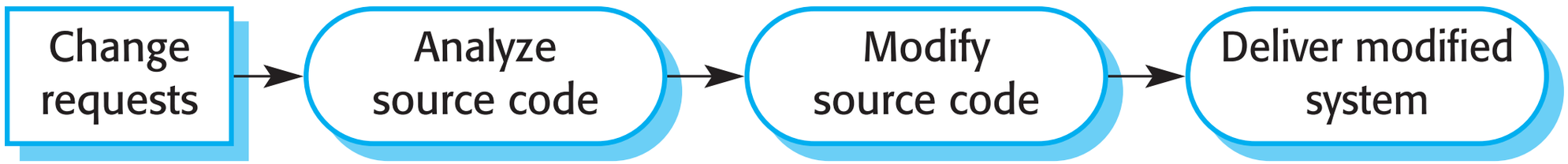
Software Evolution Process



Urgent Changes

- Urgent changes may have to be implemented without going through all stages of the software engineering process, e.g. if…
  - 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).

Urgent Changes

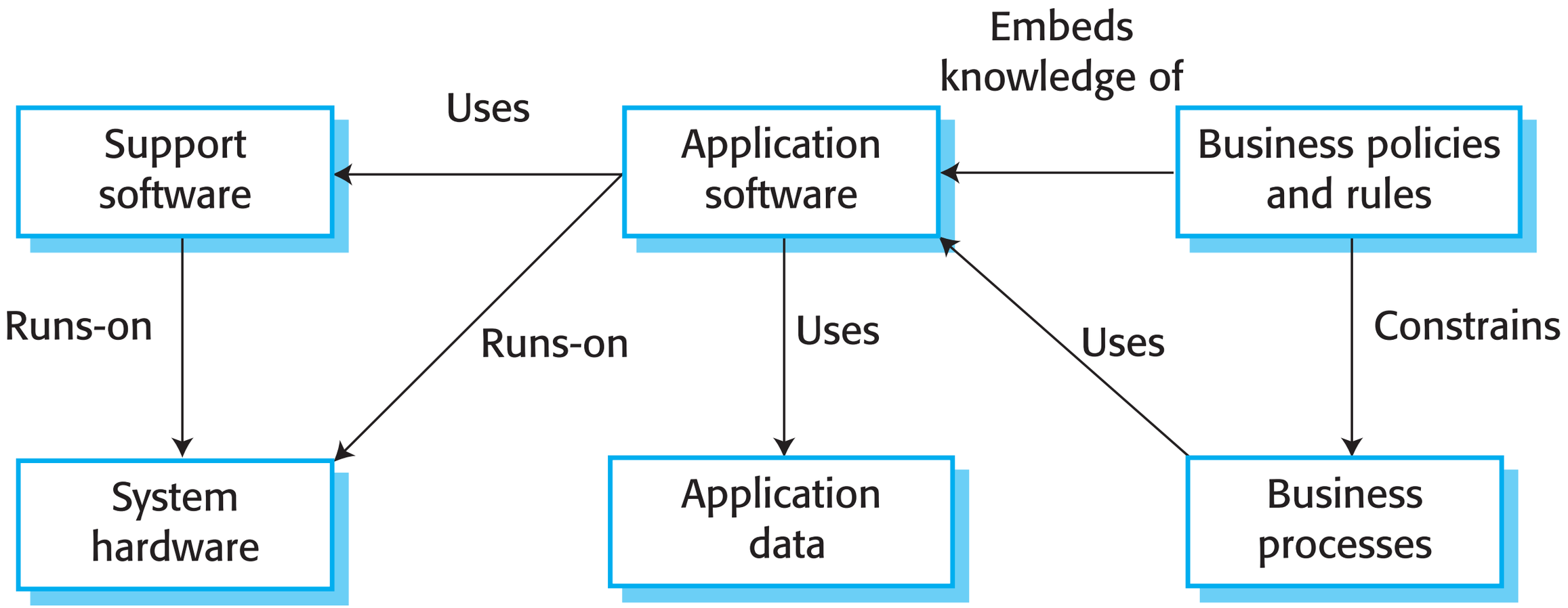


Legacy Systems

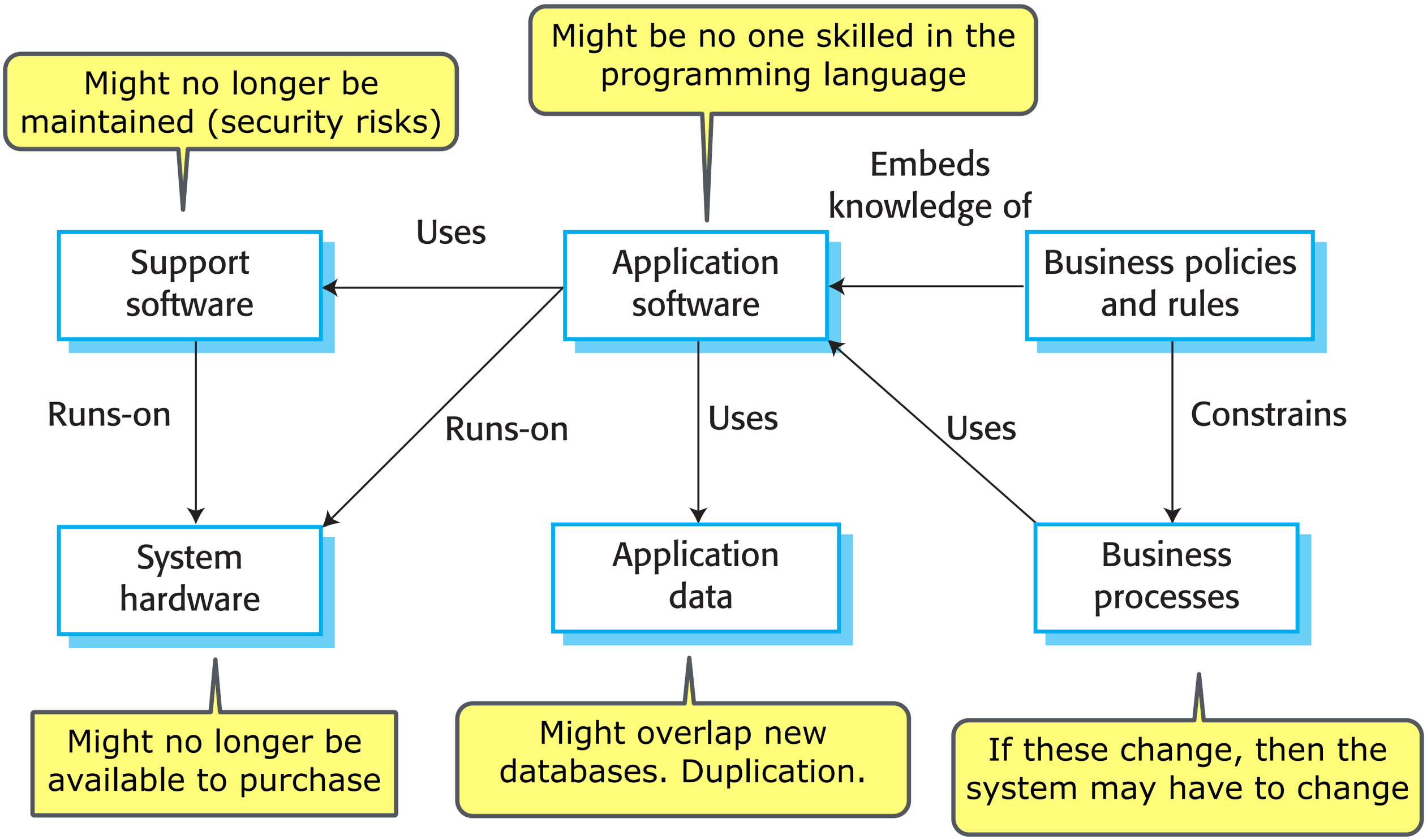
Legacy Systems

- 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.

Elements of a Legacy System



Elements of a Legacy System



Legacy System Replacement

- Legacy system replacement is risky and expensive so businesses continue to use these systems
- System replacement is risky for a number of reasons
  - Lack of complete system specification
  - Tight integration of system and business processes
  - Undocumented business rules embedded in the legacy system
  - New software development may be late and/or over budget

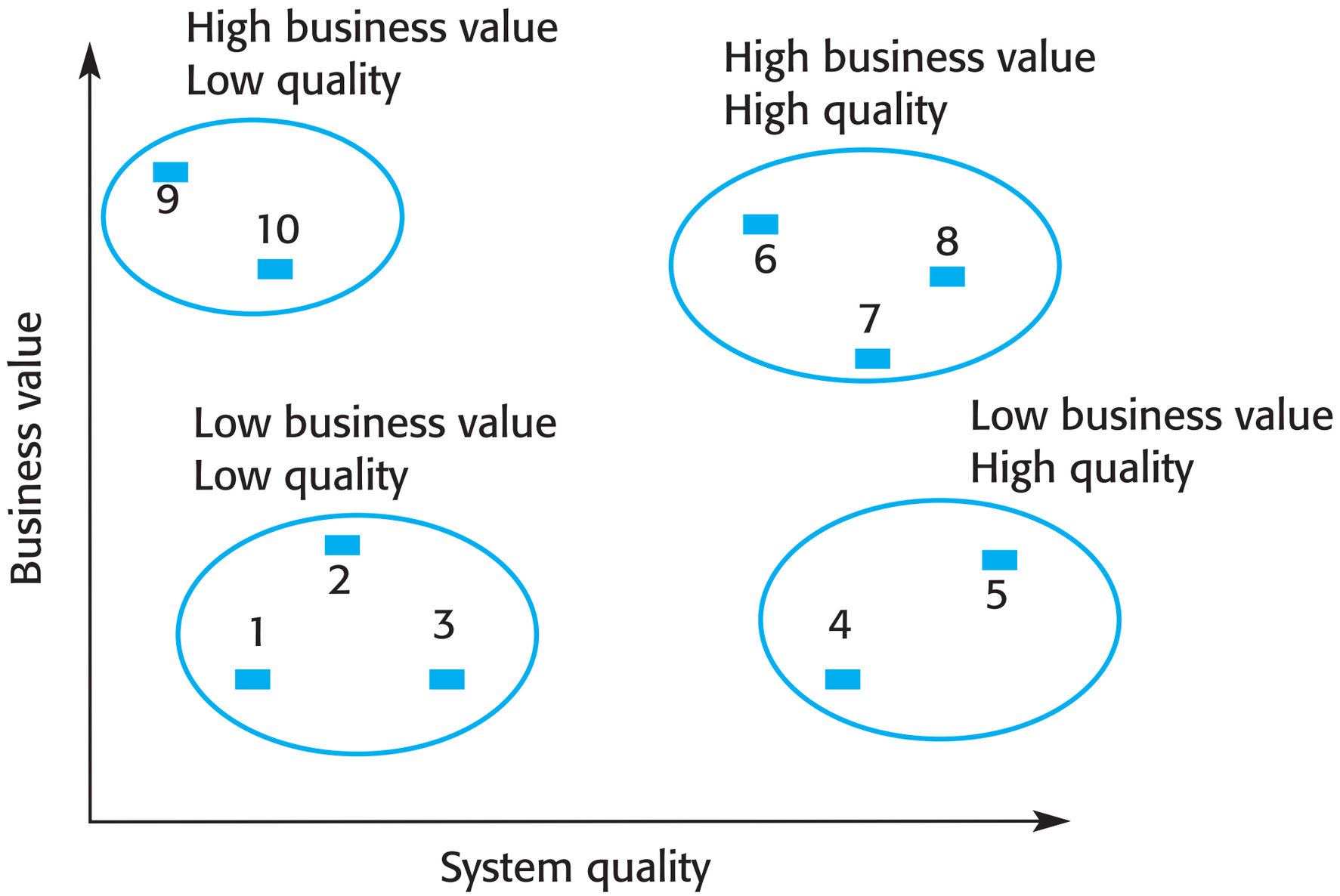
Legacy System Replacement

- Legacy systems are expensive to change for a number of reasons:
  - No consistent programming style
  - Use of obsolete programming languages with few people available with these language skills
  - Inadequate system documentation
  - System structure degradation
  - Program optimisations may make them hard to understand
  - Data errors, duplication and inconsistency

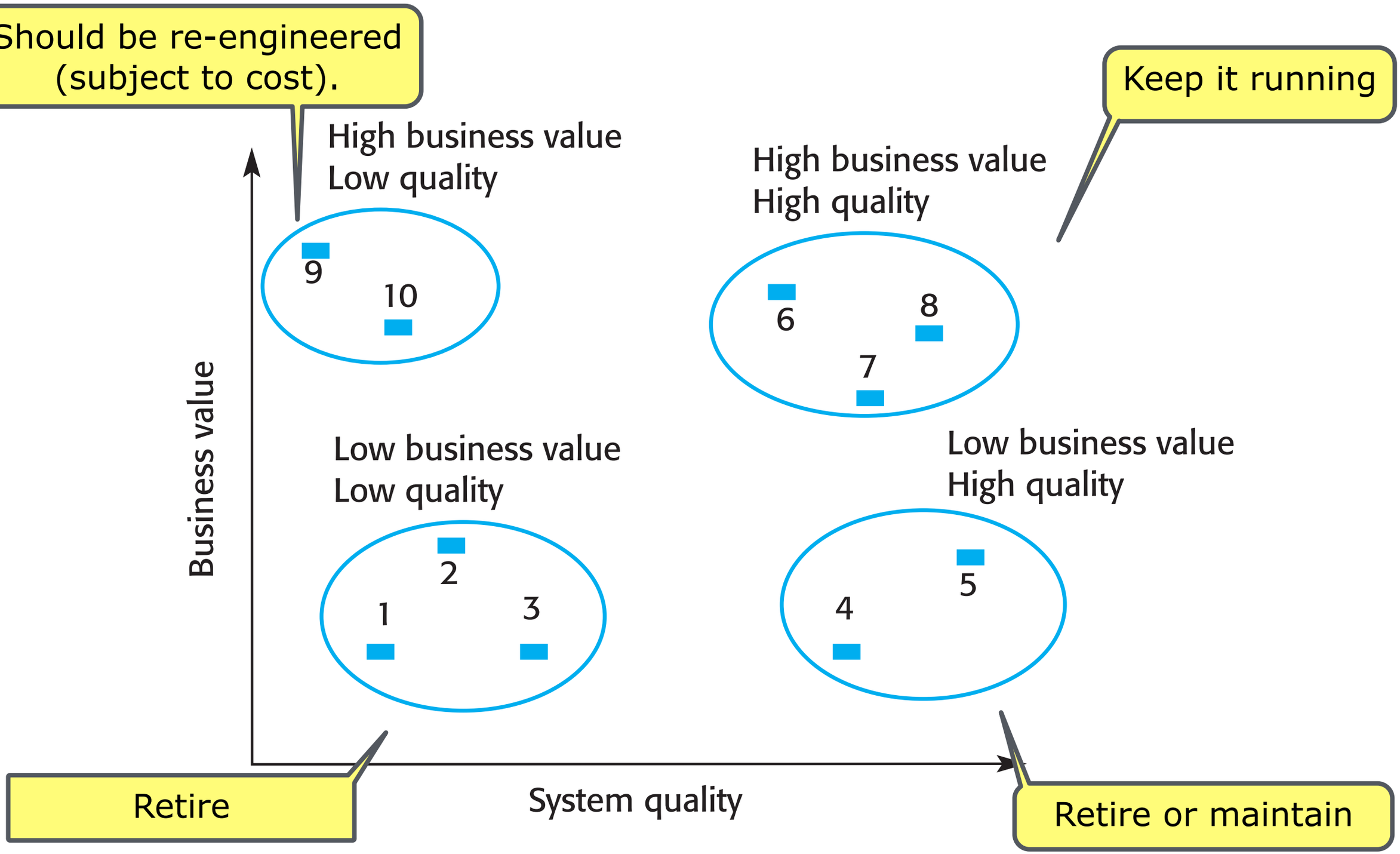
Legacy System Management

- 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.

Legacy System Management



Legacy System Management



Business Value Assessment

- Assessment should take different viewpoints into account
  - System end-users
  - Business customers
  - Line managers
  - IT managers
  - Senior managers
- Interview different stakeholders and collate results.

Business Value Assessment

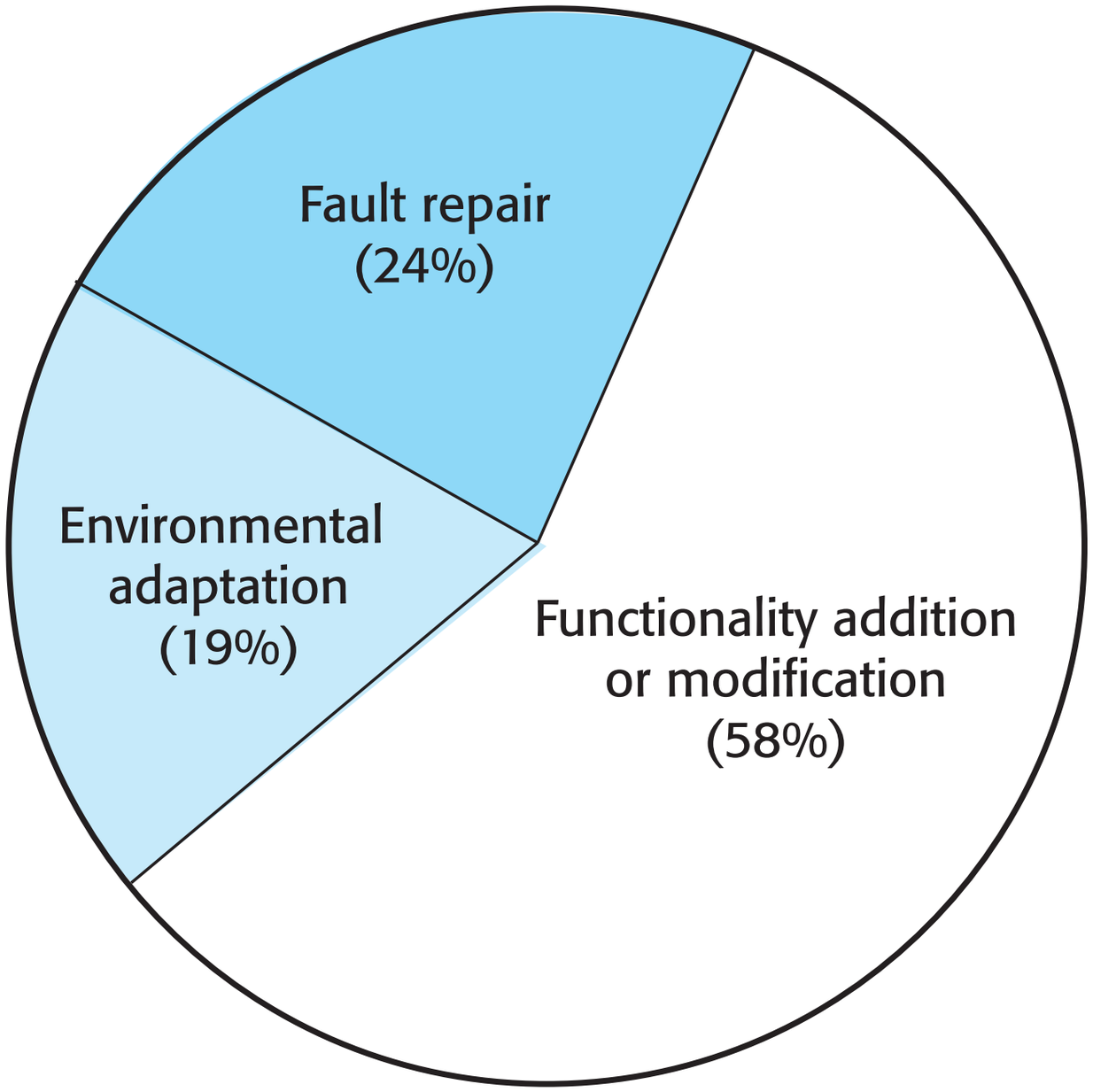
- Factors:
  - Usage: How many people use the system? How many hours per day?
  - Cost savings: Could a new system improve business process efficiency or reduce costs?
  - Maintenance costs: Could a new system reduce maintenance costs?
  - Business opportunity: Could a new system enable a new feature that creates competitive advantage?
  - Dependability: Does the legacy system have outages?
  - Performance: Is system performance adequate?
  - Interoperability? Can new systems share data with the legacy system?
  - Skills: Are the skills available to support the system?

Software Maintenance

Software Maintenance

- Modifying a program after it has been put into use.
- Does not normally involve major changes to the system architecture.
- Types:
  - Fault repair: Bug repairs and fixes for security vulnerabilities
  - Environment changes: Changes in the operating system or interacting systems
  - Modify or adding functionality to meet new requirements

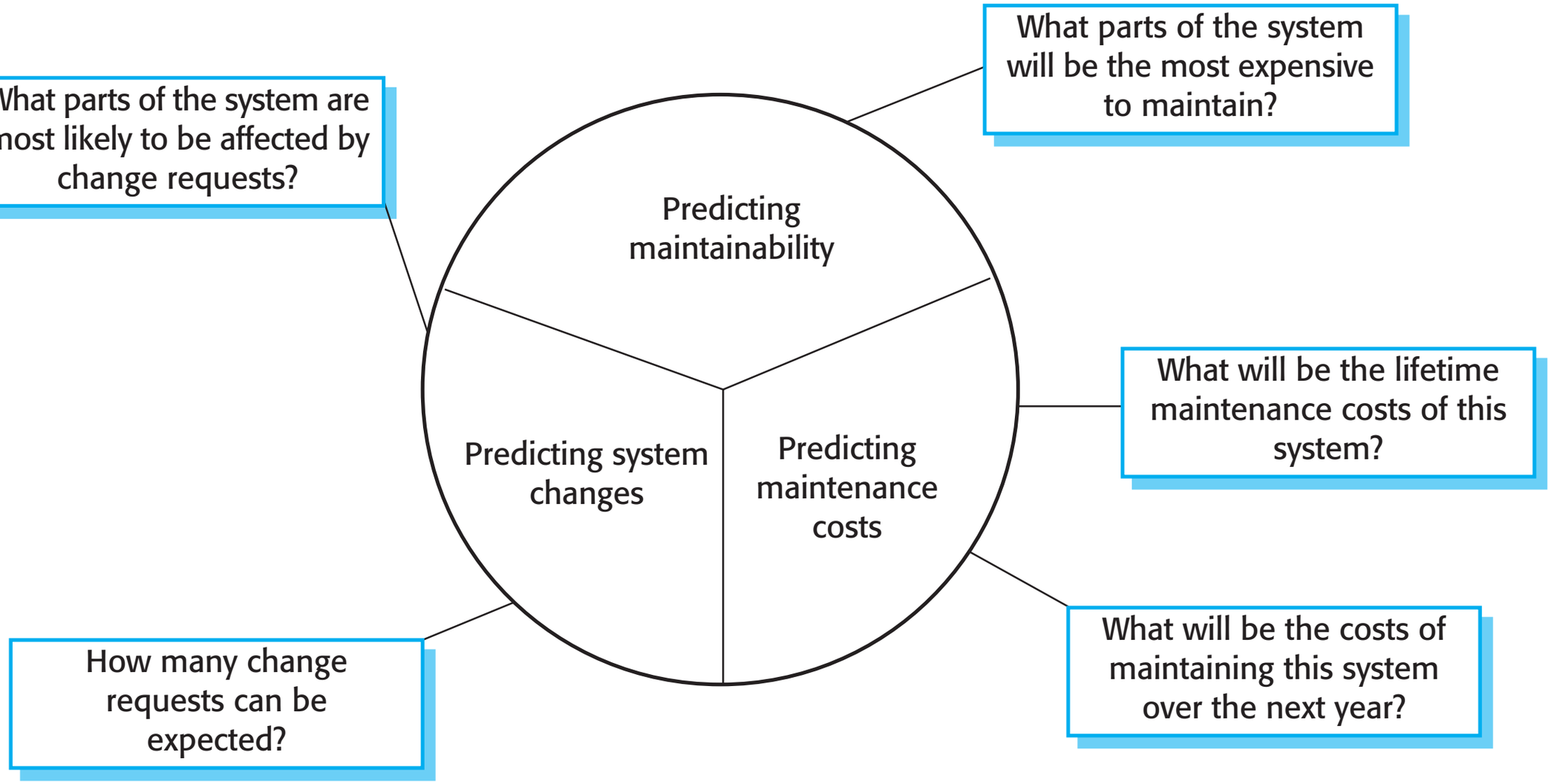
Software Maintenance



Software Maintenance Costs

- Typically, 2 - 100 times the development costs
- More expensive to add features during maintenance than in development.
- As programs age, their structure degrades and they become harder to change.

Maintenance Prediction



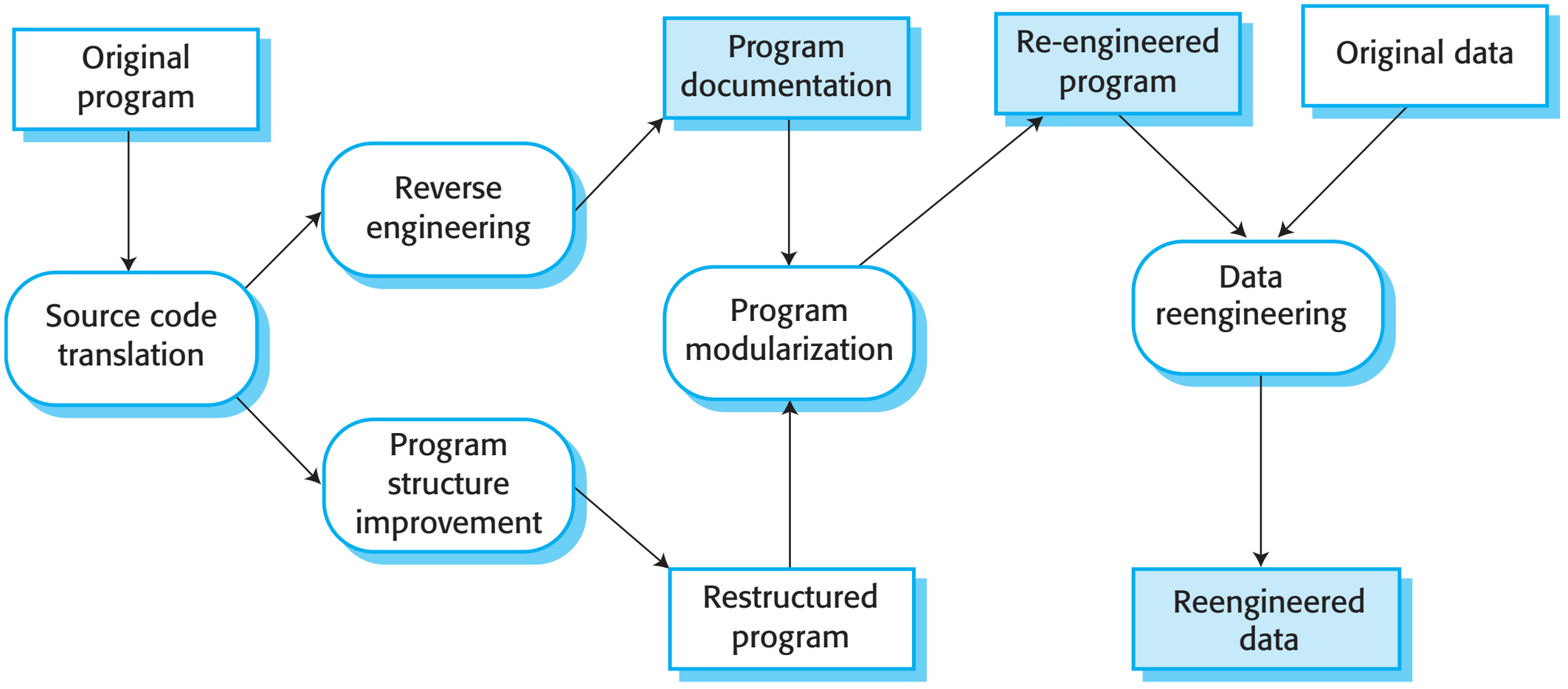
Maintenance Metrics

- Metrics:
  - Number of requests for corrective maintenance
  - Average time required for impact analysis
  - Average time taken to implement a change request
  - Number of outstanding change requests.
- If any or all of these is increasing, this may indicate a decline in maintainability.

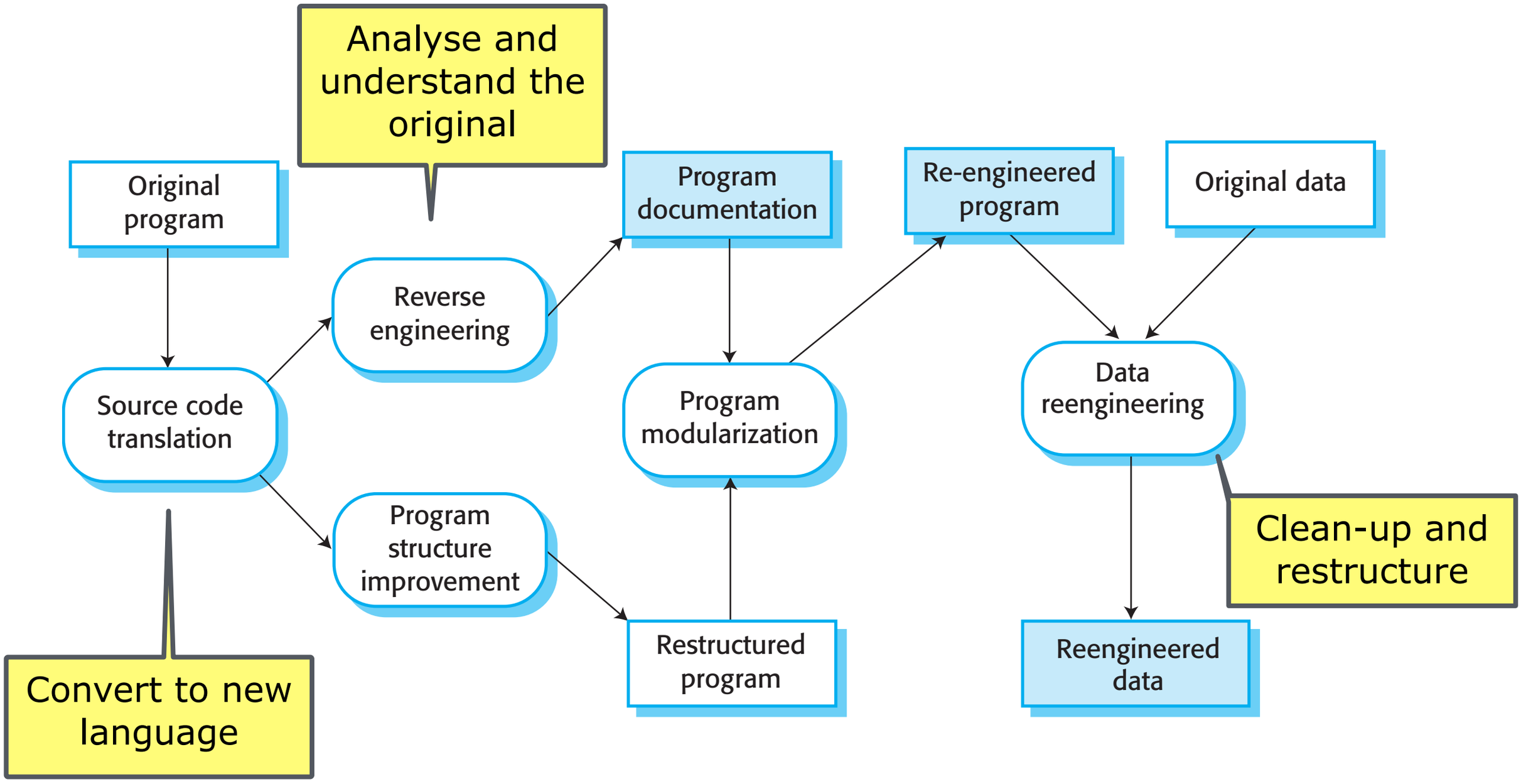
Software Re-Engineering

- … is restructuring or rewriting part or all of a legacy system without changing its functionality.
- Compared to development of wholly new systems, re-engineering is lower risk and lower cost because there is working system to refer to and to test against.
- Can run the systems in parallel for a time can check that they give the same results.

Re-Engineering Process



Re-Engineering Process



Refactoring

- … is the process of making improvements to a program to slow down degradation through change.
- When you refactor a program, you should not add or modify functionality but rather concentrate on source code improvement.
- It is a continuous process of improvement throughout the development and evolution process.
- It is not just a menu option in Eclipse…

Bad Smells

- A code smell is any characteristic in the source code of a program that possibly indicates a deeper problem.
- Kent Beck in the late 1990s.
- Refactoring: Improving the Design of Existing Code by Martin Fowler, 1999.

Bad Smells

- 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

- 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.

- 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 of the data.
- Speculative generality
  - This occurs when developers include generality in a program in case it is required in the future. This can often simply be removed.

Summary

- 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 costs 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 systems are older software systems, developed using obsolete software and hardware technologies, that remain useful for a business.
- It is often cheaper and less risky to maintain a legacy system than to develop a replacement system using modern technology.
- 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.