Software Evolution

IS301 – Software Engineering

Lecture #28 – 2004-11-05

M. E. Kabay, PhD, CISSP-ISSMP

Professor of Computer Information Systems
School of Business & Management, Norwich University

mailto:mkabay@norwich.edu **V: 802.479.7937**

Objectives

- To explain why change is inevitable if software systems are to remain useful
- To discuss software maintenance and maintenance cost factors
- To describe the processes involved in software evolution
- To discuss an approach to assessing evolution strategies for legacy systems

Topics

- Program evolution dynamics
- Software maintenance
- Evolution processes
- Legacy system evolution

Software Change (1)

- Managing processes of software system change

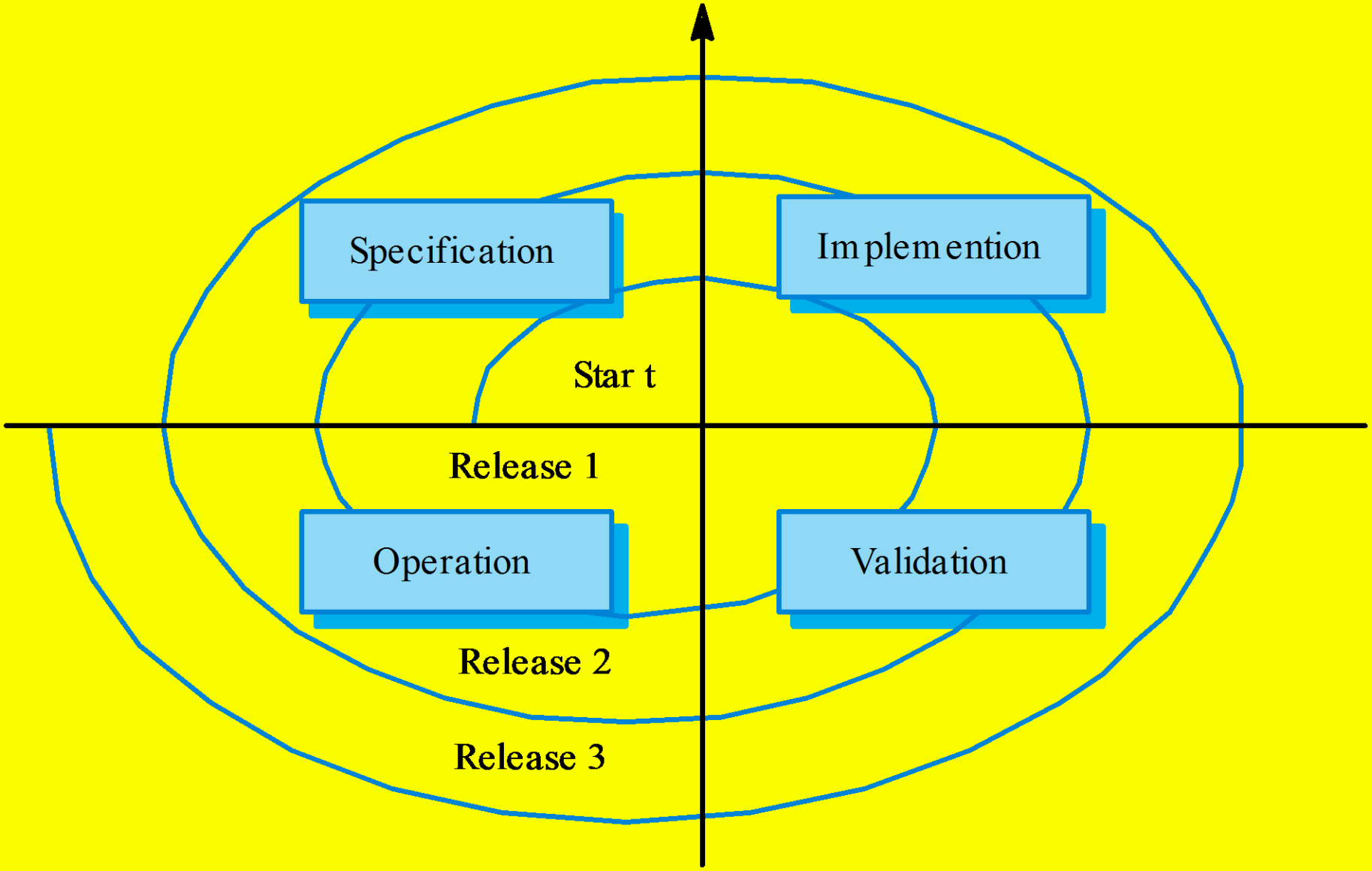
Software Change (2)

- Software change inevitable
  - New requirements emerge when software used
  - Business environment changes
  - Errors must be repaired
  - New equipment must be accommodated
  - Performance or reliability may have to be improved
- Key problem for organizations:
  - Implementing and managing change to legacy systems

Software Change Strategies

- Software maintenance
  - Response to changed requirements
  - Fundamental software structure stable
- Architectural transformation
  - Generally from centralized architecture to distributed architecture
- Software re-engineering
  - No new functionality added
  - Restructured and reorganized
  - To facilitate future changes
- Strategies may be applied separately or together

Spiral Model Of Evolution



Program Evolution Dynamics

- Study of *processes of system change*
- Lehman and Belady
  - Major empirical study
  - Proposed ‘laws’ applying to all systems as they evolved
- Sensible observations rather than *laws*
  - Applicable to large systems developed by large organizations
  - Perhaps less applicable in other cases

Lehman's Laws

- Continuing Change
- Increasing Complexity
- Large Program Evolution
- Organizational Stability
- Conservation of Familiarity

Continuing Change

- A program used in a real-world environment must necessarily change or it will progressively become less useful in that environment.

Increasing Complexity

- As an evolving program changes, its structure tends to become more complex.
- Extra resources must be devoted to preserving and simplifying the structure.

Large Program Evolution

- Program evolution is a self-regulating process.
- System attributes such as size, time between releases and the number of reported errors are approximately invariant for each system release.

Organizational Stability

- Over a program's lifetime, its rate of development is approximately constant and independent of the resources devoted to system development.

Conservation of Familiarity

- Over the lifetime of a system, the incremental change in each release is approximately constant.

Applicability of Lehman's Laws

- Not yet been established
- Generally applicable to
  - Large, tailored systems
  - Developed by large organizations
- Not clear how they should be modified for
  - Shrink-wrapped software products
  - Systems that incorporate significant number of COTS components
  - Small organizations
  - Medium sized systems

Software Maintenance

- Modifying program after it has been put into use
- Does not normally involve major changes to system's architecture
- Changes are implemented by
  - Modifying existing components and
  - Adding new components to system

Maintenance Inevitable

- System requirements likely to change while system being developed
  - Because environment changing
  - Therefore delivered system won't meet its requirements (!)
- Systems tightly coupled with their environment
  - When system installed in environment it changes that environment
  - Therefore changes system requirements
- Systems MUST be maintained if they are to remain useful in their environment

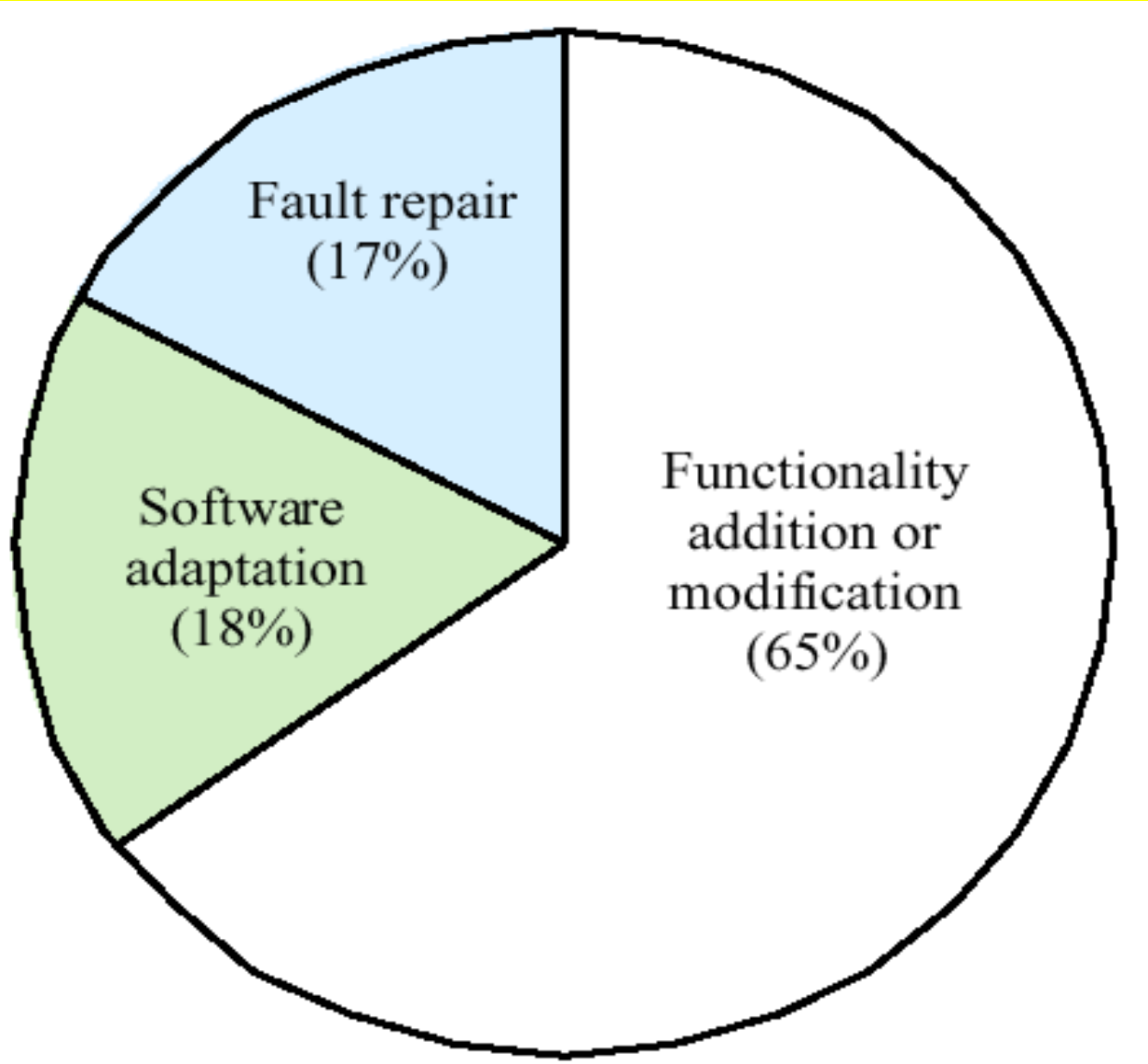
Tool/Problem Relation

Availability of a tool changes the perception of what is possible

Types of Maintenance

- Repair software faults
- Adapt software to different operating environment (e.g., new computer, OS)
- Add to or modify system's functionality

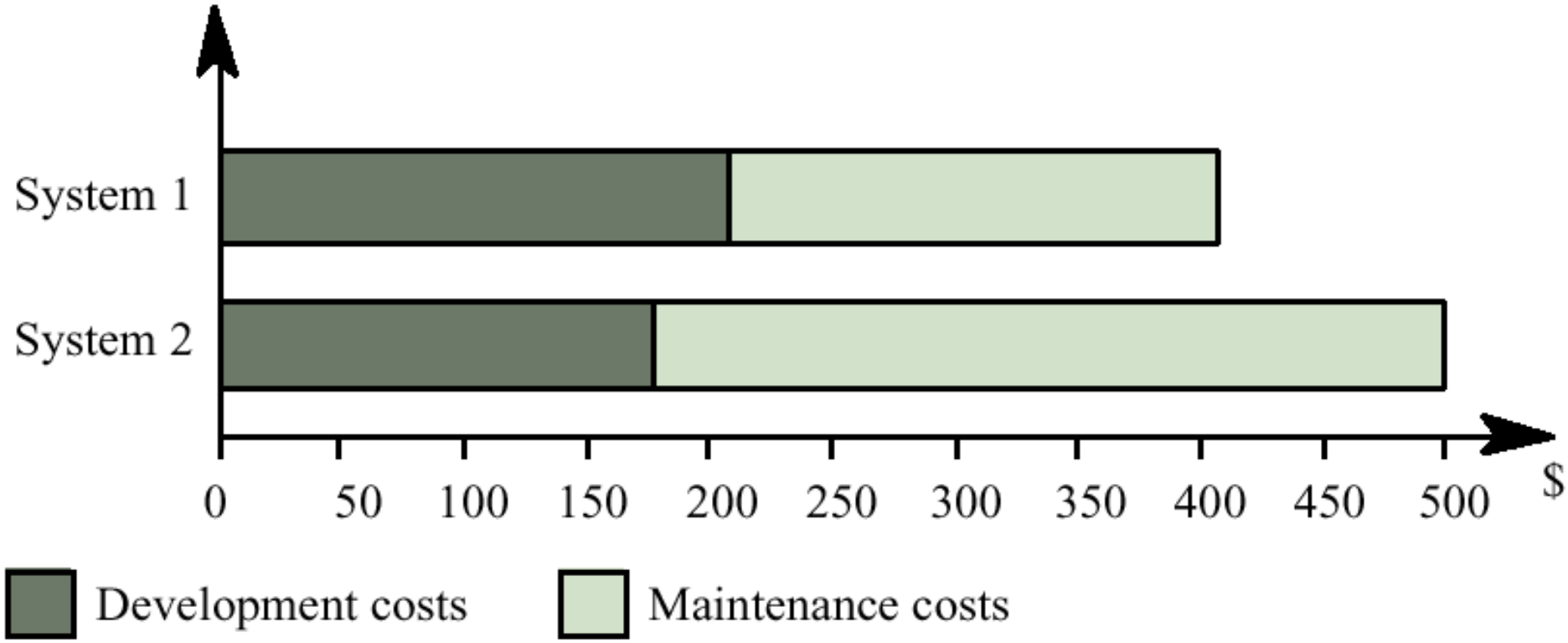
Distribution of Maintenance Effort



Maintenance Costs

- Usually greater than development costs (2* to 100* depending on application)
- Affected by both technical and non-technical factors
- Increases as software maintained
  - Maintenance corrupts software structure thus making further maintenance more difficult
- Ageing software can have high support costs (e.g. old languages, compilers etc.)

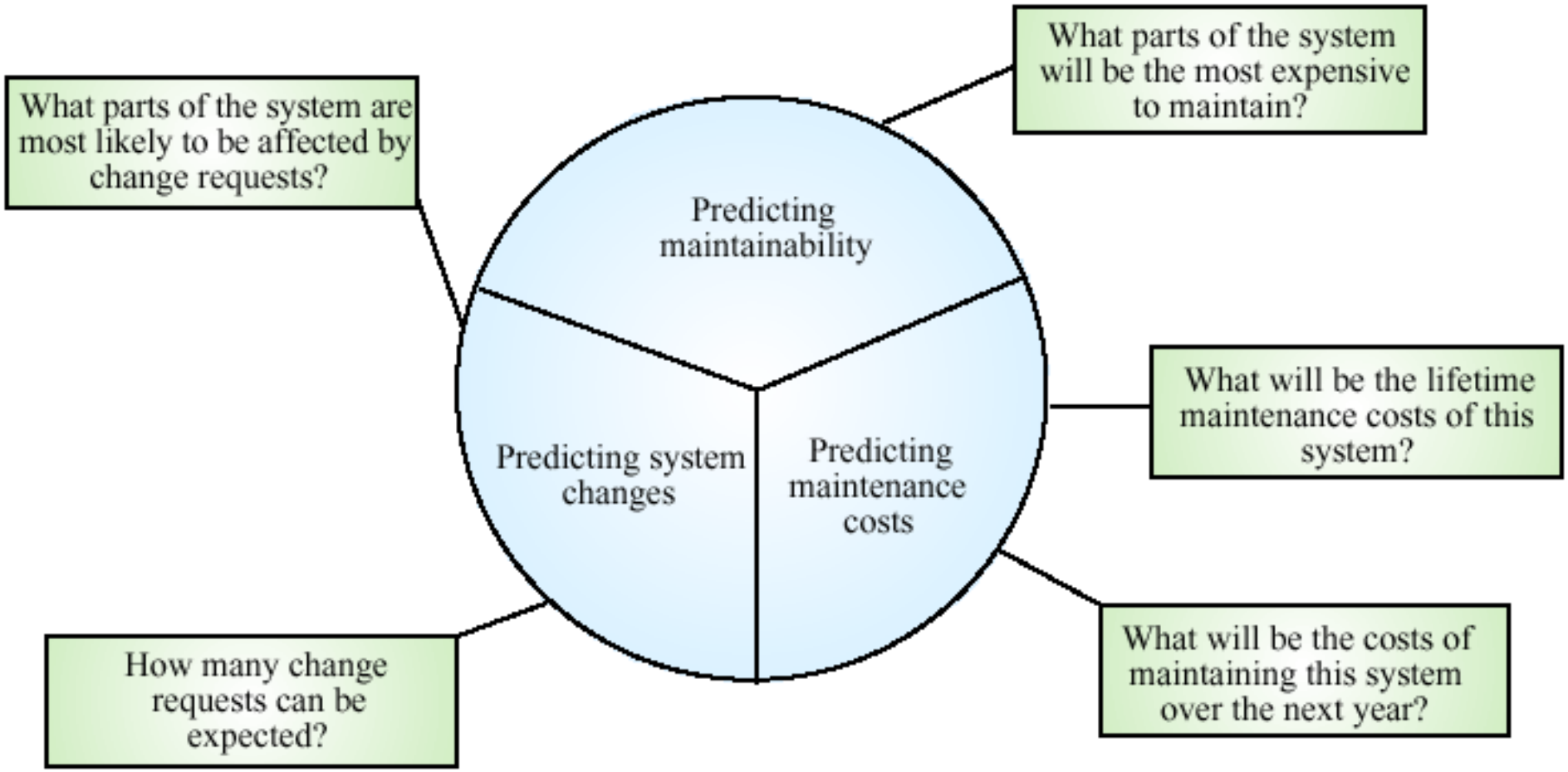
Development/Maintenance Costs



Maintenance Cost Factors

- Team stability
  - $$ reduced if same staff involved with them for some time
- Contractual responsibility
  - Developers of system may have no contractual responsibility for maintenance
  - So no incentive to design for future change
- Staff skills
  - Maintenance staff often inexperienced and may have limited domain knowledge
- Program age and structure
  - As programs age, their structure degraded and they become harder to understand and change

Maintenance Prediction



Complexity Metrics

- Predictions of maintainability can be made by assessing complexity of system components
- Studies have shown that most maintenance effort spent on relatively small number of system components
- Complexity depends on
  - Complexity of control structures
  - Complexity of data structures
  - Procedure and module size

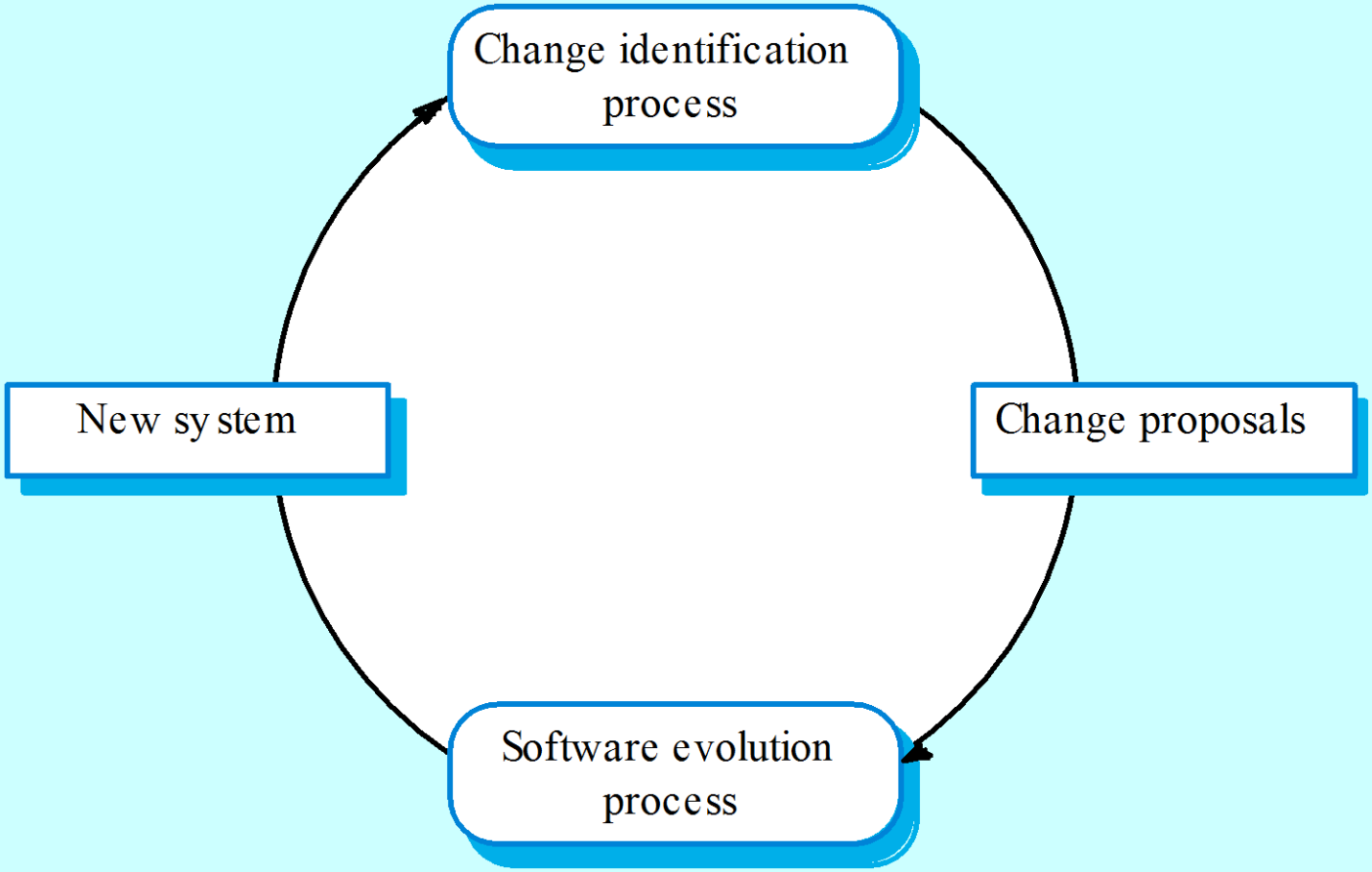
Process Metrics

- Process measurements may be used to assess maintainability
  - Number of requests for corrective maintenance
  - Average time required for impact analysis
  - Average time taken to implement change request
  - Number of outstanding change requests
- If any or all of these increasing, this may indicate decline in maintainability

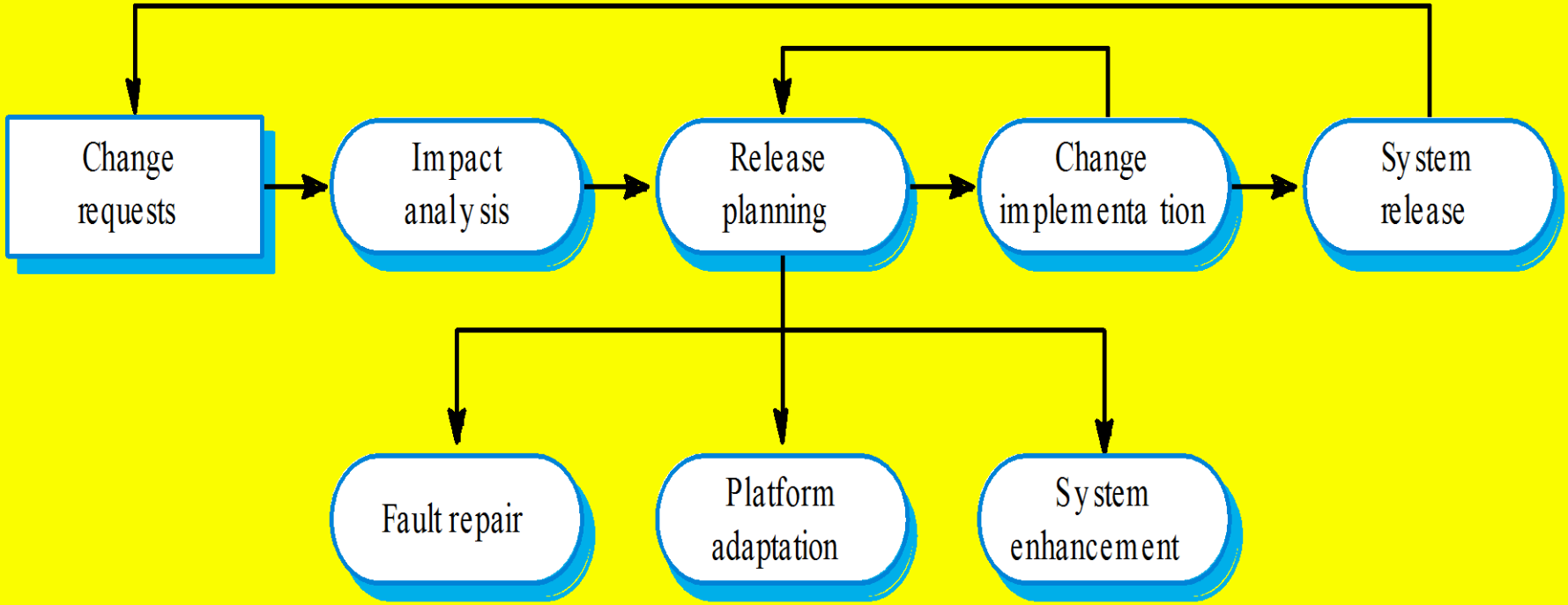
Evolution processes

- 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. Change identification and evolution continue throughout the system lifetime.

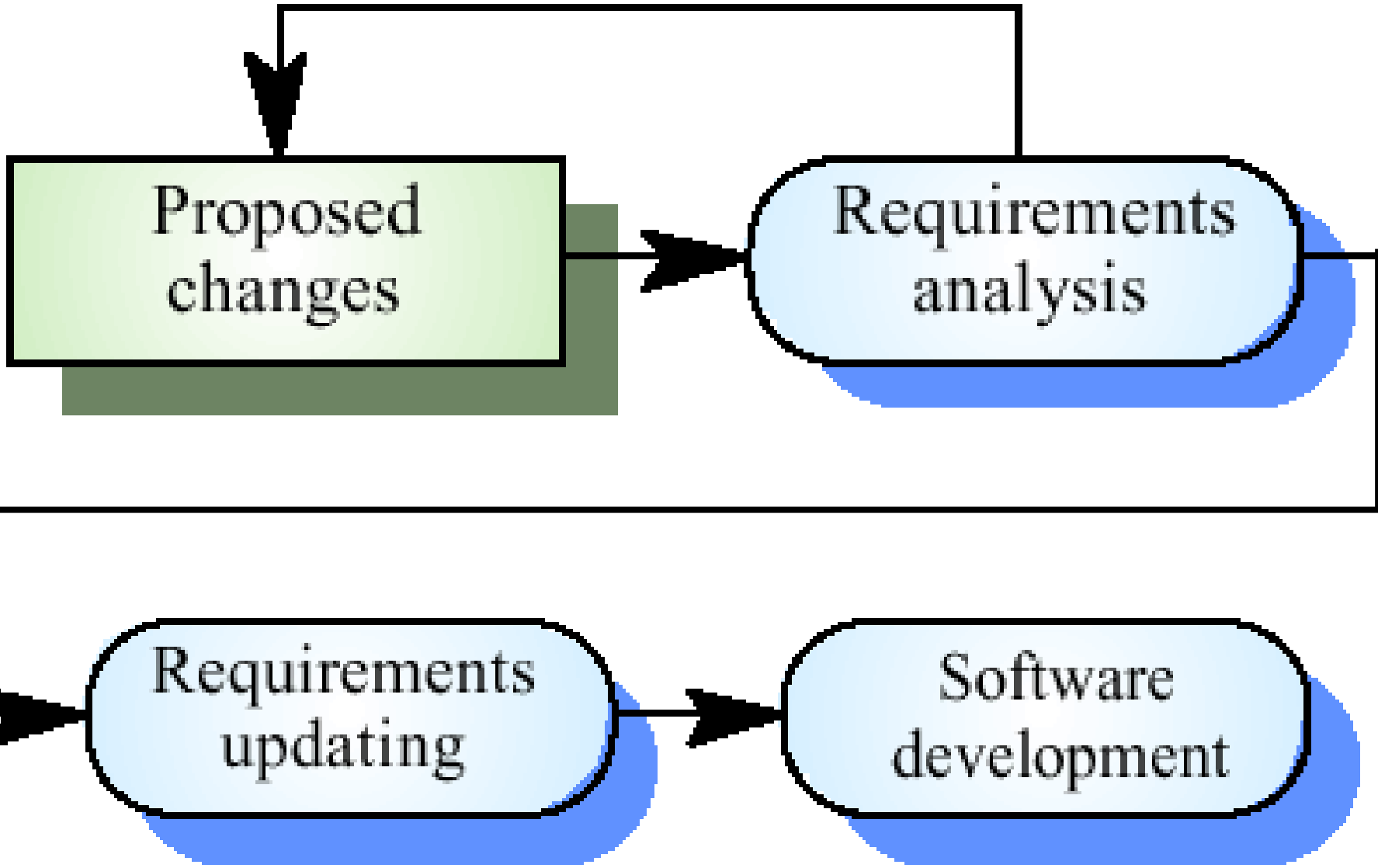
Change Identification and Evolution



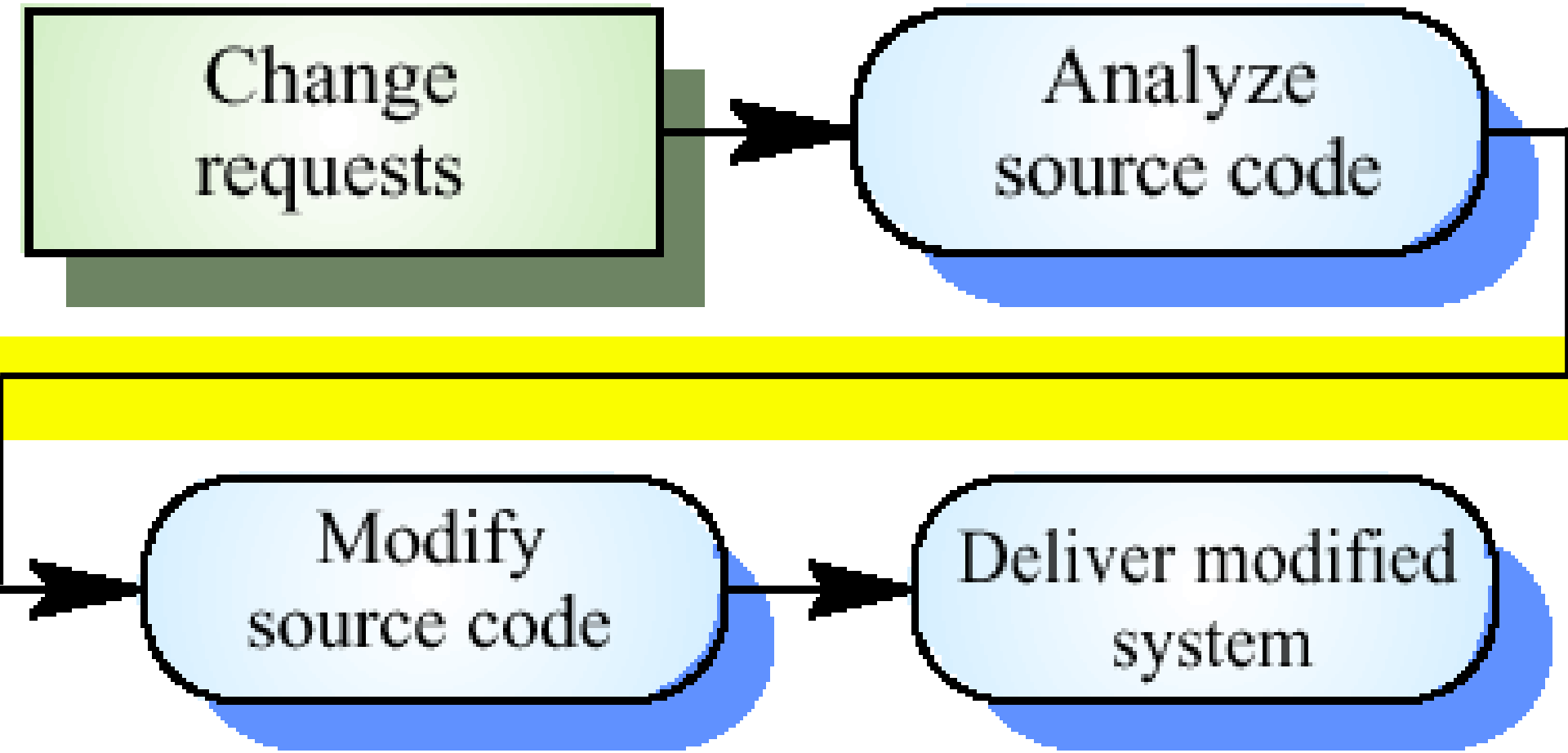
The System Evolution Process



Change Implementation



Emergency Repair



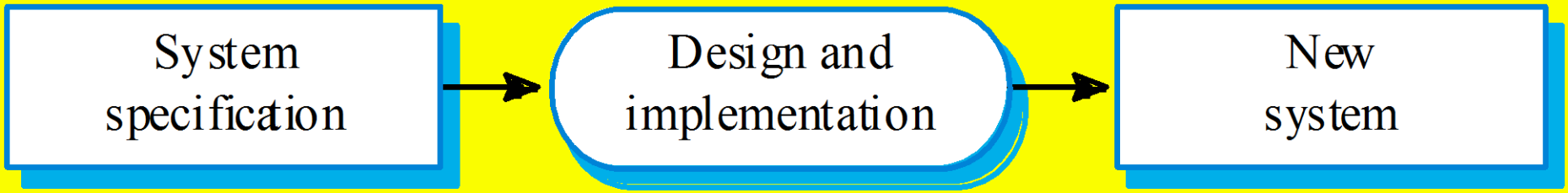
System re-engineering

- Re-structuring or re-writing part or all of a legacy system without changing its functionality.
- Applicable where some but not all sub-systems of a larger system require frequent maintenance.
- Re-engineering involves adding effort to make them easier to maintain. The system may be re-structured and re-documented.

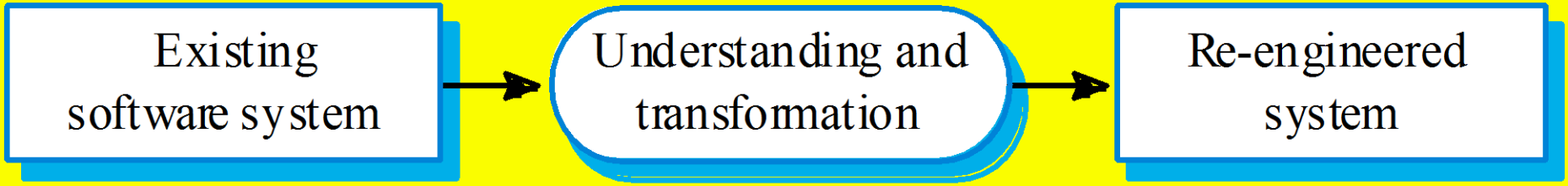
Advantages of Reengineering

- **Reduced risk**
  - **There is a high risk in new software development. There may be development problems, staffing problems and specification problems.**
- **Reduced cost**
  - **The cost of re-engineering is often significantly less than the costs of developing new software.**

Forward and Re-engineering

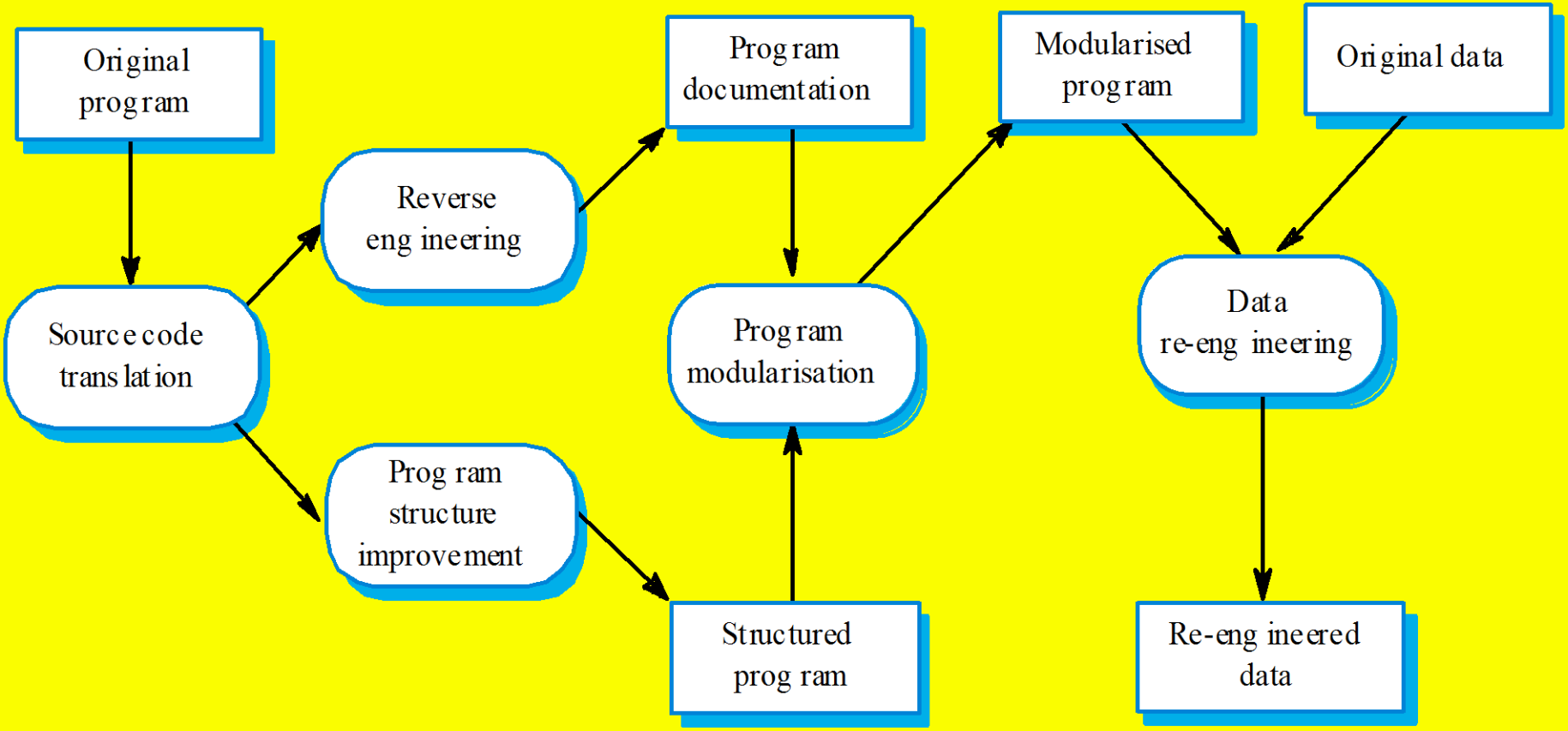


Forward engineering



Software re-engineering

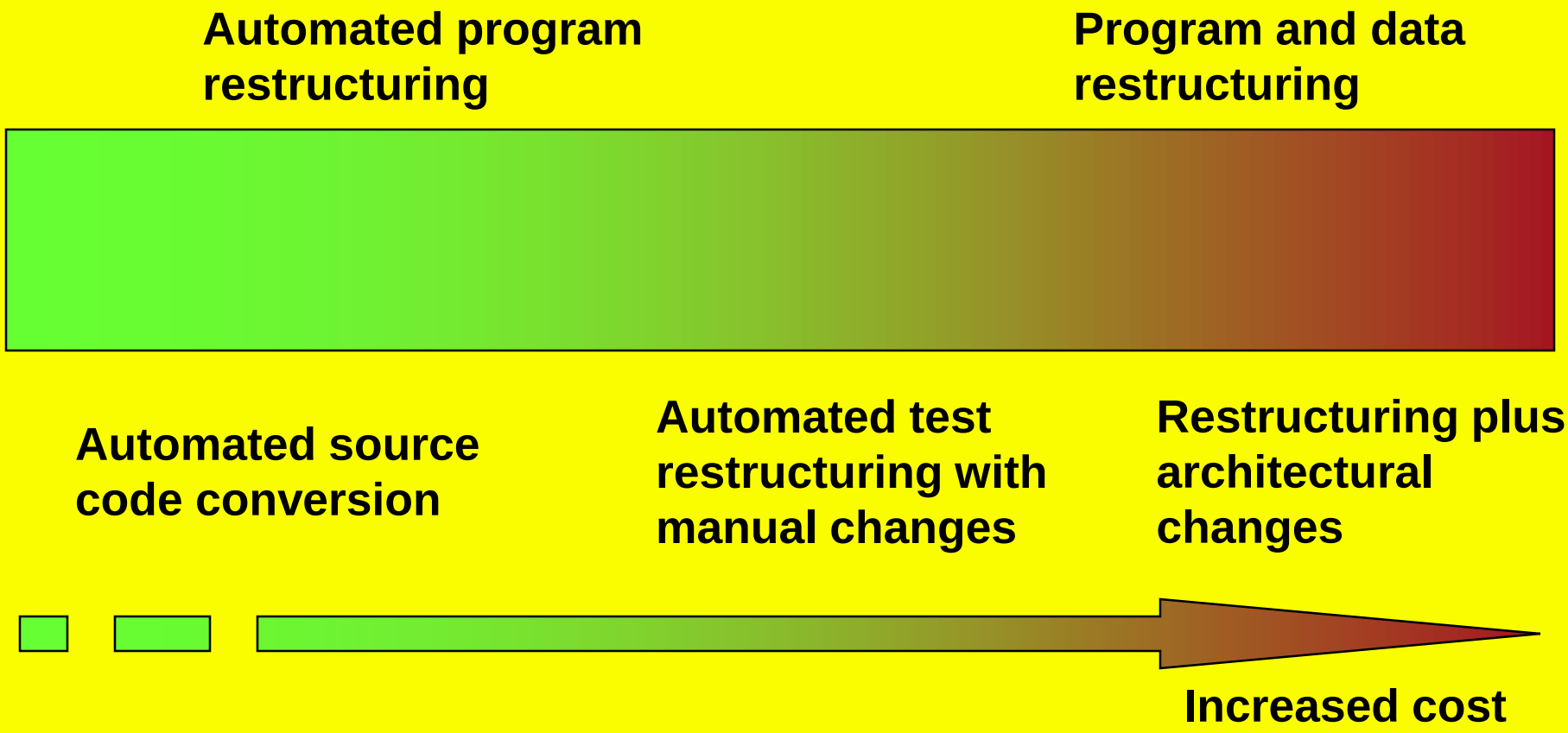
The re-engineering process



Reengineering Process Activities

- Source code translation
  - Convert code to a new language.
- Reverse engineering
  - Analyze the program to understand it;
- Program structure improvement
  - Restructure automatically for understandability;
- Program modularization
  - Reorganize the program structure;
- Data reengineering
  - Clean-up and restructure system data.

Re-engineering Approaches



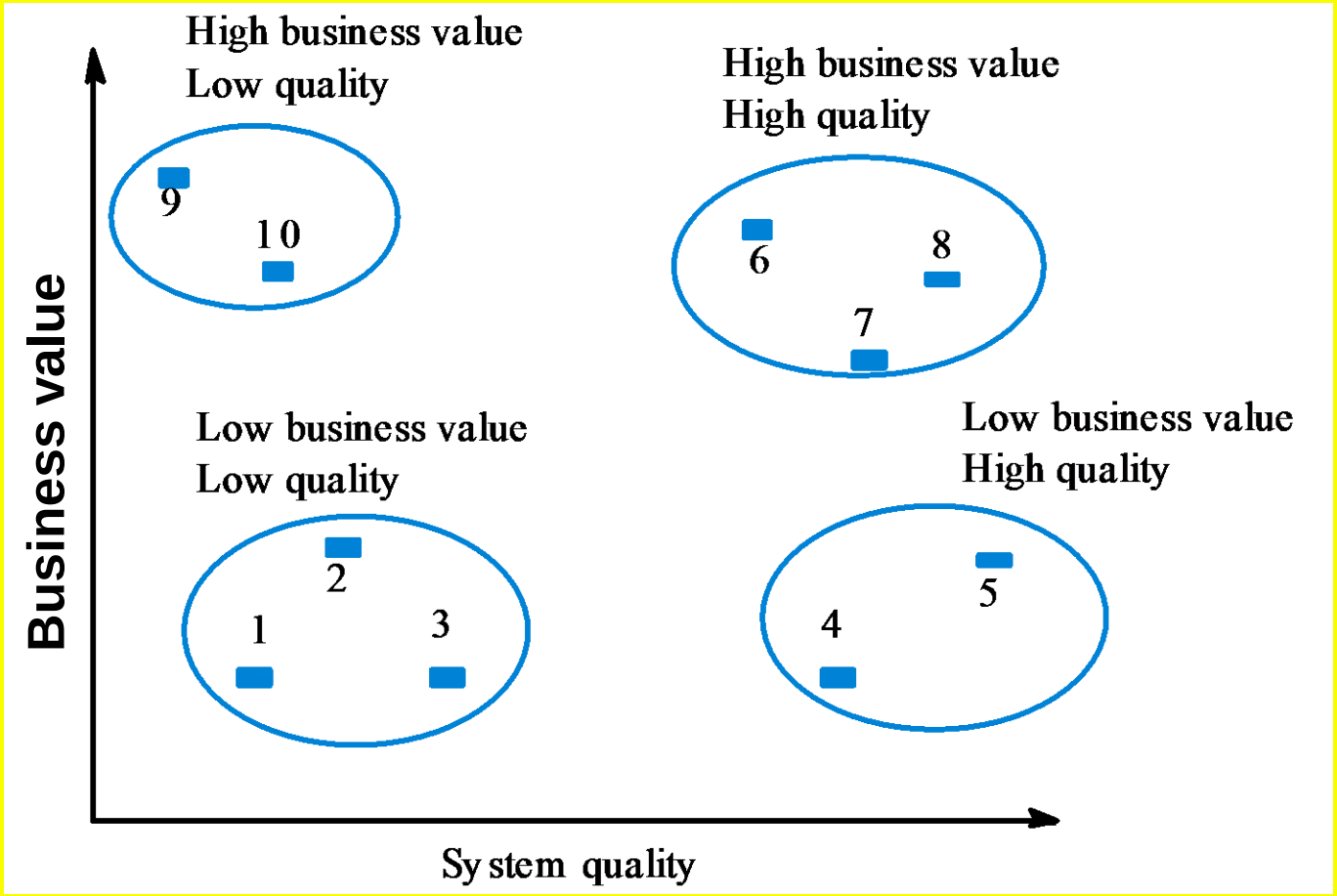
Reengineering Cost Factors

- The quality of the software to be reengineered.
- The tool support available for reengineering.
- The extent of the data conversion which is required.
- The availability of expert staff for reengineering.
  - This can be a problem with old systems based on technology that is no longer widely used.

Legacy System Evolution

- Organizations 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.

System Quality and Business Value



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.

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.

System Quality Assessment

- Business process assessment
  - How well does the business process support the current goals of the business?
- Environment assessment
  - How effective is the system's environment and how expensive is it to maintain?
- Application assessment
  - What is the quality of the application software system?

Business Process Assessment

- Use a viewpoint-oriented approach and seek answers from system stakeholders
  - Is there a defined process model and is it followed?
  - Do different parts of the organization use different processes for the same function?
  - How has the process been adapted?
  - What are the relationships with other business processes and are these necessary?
  - Is the process effectively supported by the legacy application software?
- Example - a travel-office system may now have a low business value because of the widespread use of Web-based ordering.

Environment Assessment (1)

- Supplier stability
  - Is the supplier is still in existence?
  - Is the supplier financially stable and likely to continue in existence?
  - If the supplier is no longer in business, does someone else maintain the systems?
- Failure rate
  - Does the hardware have a high rate of reported failures?
  - Does the support software crash and force system restarts?

Environment Assessment (2)

- Age
  - How old is the hardware and software?
    - ✓The older the hardware and support software, the more obsolete it will be.
    - ✓It may still function correctly but there could be significant economic and business benefits to moving to more modern systems.
  - Performance
    - ✓Is the performance of the system adequate?
    - ✓Do performance problems have a significant effect on system users?

Environment Assessment (3)

- Support requirements
  - What local support is required by the hardware and software?
  - If there are high costs associated with this support, it may be worth considering system replacement.
- Maintenance costs
  - What are the costs of hardware maintenance and support software licences?
  - Older hardware may have higher maintenance costs than modern systems.
  - Support software may have high annual licensing costs.

Environment Assessment (4)

- Interoperability
  - Are there problems interfacing the system to other systems?
  - Can compilers etc. be used with current versions of the operating system?
  - Is hardware emulation required?

Application Assessment (1)

- Support requirements
  - What local support is required by the hardware and software?
  - If there are high costs associated with this support, it may be worth considering system replacement.
- Maintenance costs
  - What are the costs of hardware maintenance and support software licences?
  - Older hardware may have higher maintenance costs than modern systems.
  - Support software may have high annual licensing costs.

Application Assessment (2)

- Interoperability
  - Are there problems interfacing the system to other systems?
  - Can compilers etc. be used with current versions of the operating system?
  - Is hardware emulation required?
- Programming language
  - Are modern compilers available for the programming language used to develop the system?
  - Is the programming language still used for new system development?

Application Assessment (3)

- Configuration management
  - Are all versions of all parts of the system managed by a configuration management system?
  - Is there an explicit description of the versions of components that are used in the current system?
- Test data
  - Do test data for the system exist?
  - Is there a record of regression tests carried out when new features have been added to the system?
- Personnel skills
  - Are there people available who have the skills to maintain the application?
  - Are there only a limited number of people who understand the system?

System Measurement

- You may collect quantitative data to make an assessment of the quality of the application system
  - The number of system change requests;
  - The number of different user interfaces used by the system;
  - The volume of data used by the system.

Key points

- Software development and evolution should be a single iterative process.
- Lehman's Laws describe a number of insights into system evolution.
- Three types of maintenance are bug fixing, modifying software for a new environment and implementing new requirements.
- For custom systems, maintenance costs usually exceed development costs.

Key points

- The process of evolution is driven by requests for changes from system stakeholders.
- Software re-engineering is concerned with re-structuring and re-documenting software to make it easier to change.
- The business value of a legacy system and its quality should determine the evolution strategy that is used.

Now go and study