

Software Engineering

Software Evolution

Based on Software Engineering, 7th Edition by Ian Sommerville

Stan Kurkovsky

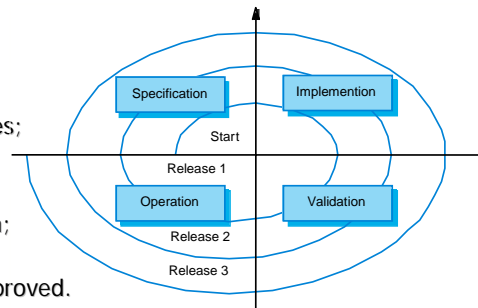
Objectives

- 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

Stan Kurkovsky

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 is added to the system;
 - The performance or reliability of the system may have to be improved.



- A key problem for organisations is implementing and managing change to their existing software systems.
- 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 evolving existing software rather than developing new software.

Stan Kurkovsky

Program evolution dynamics

- **Program evolution dynamics** is the study of the processes of system change.
- After major empirical studies, Lehman and Belady proposed that there were a number of 'laws' which applied to all systems as they evolved.
- There are sensible observations rather than laws. They are applicable to large systems developed by large organisations. Perhaps less applicable in other cases.

| Law | Description |
|-----------------------------|---|
| Continuing change | A program that is used in a real-world environment necessarily must change or become progressively 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 is approximately invariant for each system release. |
| Organisational 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. |
| Continuing growth | The functionality offered by systems has to continually increase to maintain user satisfaction. |
| Declining quality | The quality of systems will appear to be declining unless they are adapted to changes in their operational environment. |
| Feedback system | Evolution processes incorporate multi-agent, multi-loop feedback systems and you have to treat them as feedback systems to achieve significant product improvement. |

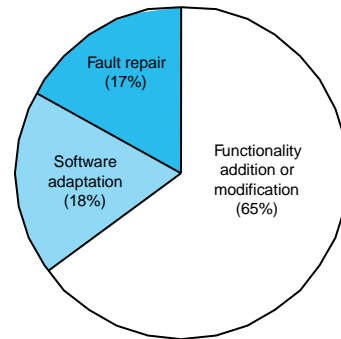
Stan Kurkovsky

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.

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.



Stan Kurkovsky

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.).

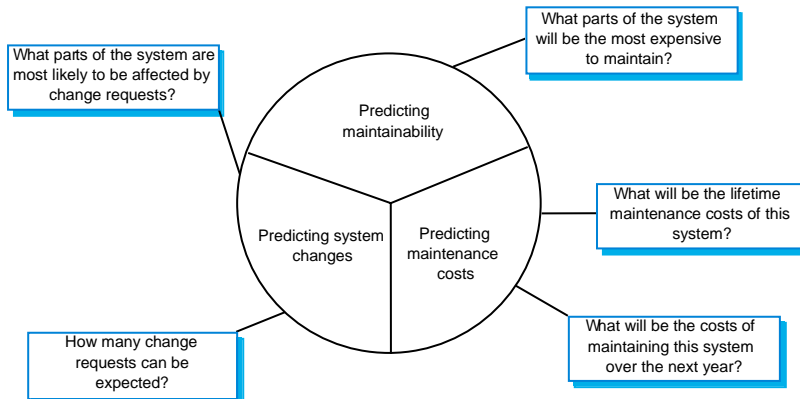
Cost factors

- 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.

Stan Kurkovsky

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.



Stan Kurkovsky

Change prediction

- Predicting the number of changes requires an understanding of the relationships between a system and its environment.
- Tightly coupled systems require changes whenever the environment is changed.
- Factors influencing this relationship are
 - Number and complexity of system interfaces;
 - Number of inherently volatile system requirements;
 - The business processes where the system is used.

Complexity metrics

- Predictions of maintainability can be made by assessing the complexity of system components.
- Studies have shown that most maintenance effort is spent on a relatively small number of system components.
- Complexity depends on
 - Complexity of control structures;
 - Complexity of data structures;
 - Object, method (procedure) and module size.

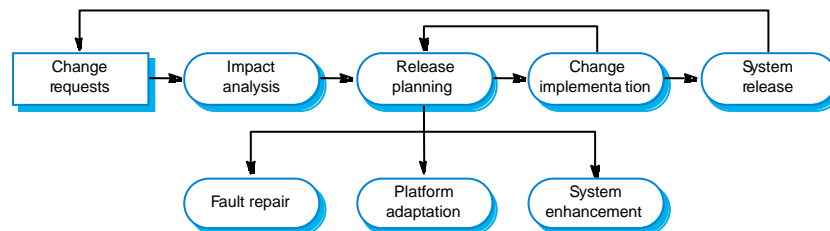
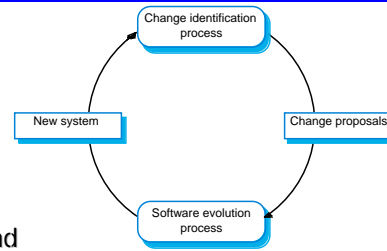
Process metrics

- 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 a change request;
 - Number of outstanding change requests.
- If any or all of these is increasing, this may indicate a decline in maintainability.

Stan Kurkovsky

Evolution processes

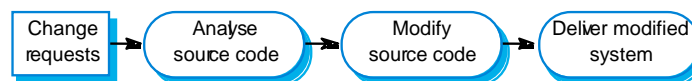
- Evolution processes depend on
 - The type of software being maintained;
 - The development processes used;
 - The skills and experience of the people involved.
- Proposals for change are the driver for system evolution. Change identification and evolution continue throughout the system lifetime.



Stan Kurkovsky

Urgent change requests

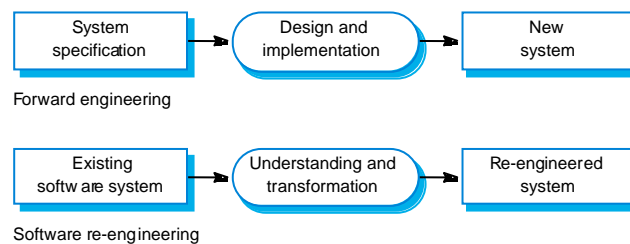
- Urgent changes may have to be implemented without going through all stages of the software engineering process
 - If a serious system fault has to be repaired;
 - If changes to the system's environment (e.g. an OS upgrade) have unexpected effects;
 - If there are business changes that require a very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



Stan Kurkovsky

System re-engineering

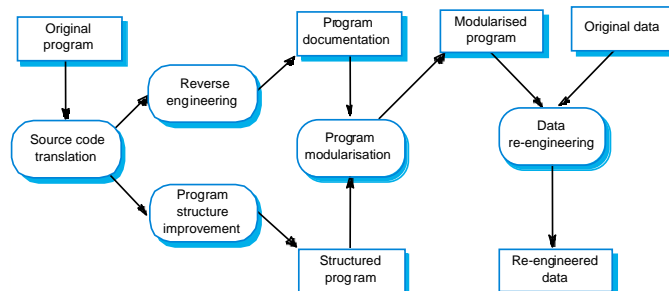
- Re-structuring or re-writing part or all of a legacy system without changing its functionality.
- Applicable where some but not all sub-systems of a larger system require frequent maintenance.
- Re-engineering involves adding effort to make them easier to maintain. The system may be re-structured and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



Stan Kurkovsky

The re-engineering process

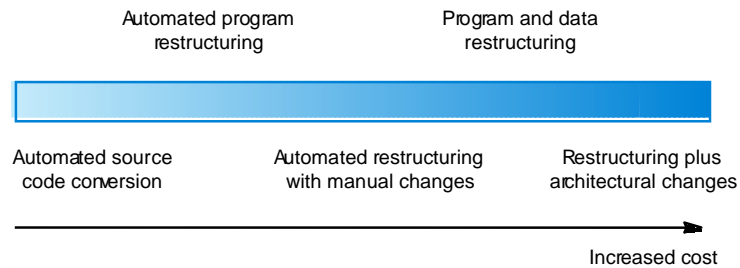
- Source code translation
 - Convert code to a new language.
- Reverse engineering
 - Analyse the program to understand it;
- Program structure improvement
 - Restructure automatically for understandability;
- Program modularisation
 - Reorganise the program structure;
- Data reengineering
 - Clean-up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



Stan Kurkovsky

Reengineering cost factors

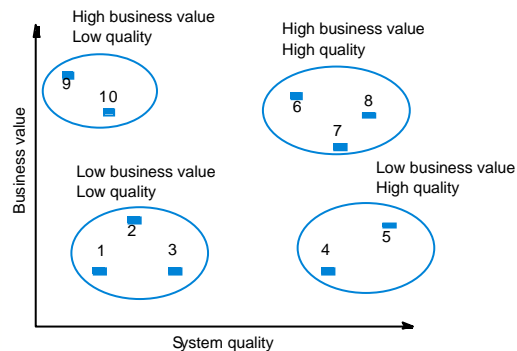
- The quality of the software to be reengineered.
- The tool support available for reengineering.
- The extent of the data conversion which is required.
- The availability of expert staff for reengineering.
 - This can be a problem with old systems based on technology that is no longer widely used.



Stan Kurkovsky

Legacy system evolution

- 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 (1,2,3);
 - Continue maintaining the system (6,7,8);
 - Transform the system by re-engineering to improve its maintainability (9,10);
 - Replace the system with a new system (4,5).
- The strategy chosen should depend on the system quality and its business value.



Stan Kurkovsky

Legacy system assessment

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?

Stan Kurkovsky

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 organisation 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 ordering system may have a low business value because of the widespread use of web-based ordering.

Stan Kurkovsky

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.
- 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.