Software Evolution

Contents

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

Introduction

- Software development does not stop when a system is delivered but continues throughout the lifetime of the system
- After a system has been deployed, it inevitably has to change if it is to remain useful
- Business changes and changes to user expectations generate new requirements for the existing software
- Parts of the software may have to be modified to correct errors that are found in operation, to adapt it for changes to its hardware and software platform, and to improve its performance or other non-functional characteristic

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 organizations is implementing and managing change to their existing software systems.

Software Evolution Importance

- Software evolution is important because organizations have invested large amounts of money in their software and are now completely dependent on these systems.
- Their systems are critical business assets and they have to invest in system change to maintain the value of these assets.
- Consequently, most large companies spend more on maintaining existing systems than on new systems development.
- Based on an informal industry poll, Erlikh (2000) suggests that 85–90% of organizational software costs are evolution costs.
- Other surveys suggest that about two-thirds of software costs are evolution costs. For sure, the costs of software change are a large part of the IT budget for all companies.

Software Evolution Importance

- Useful software systems often have a very long lifetime.
- For example, large military or infrastructure systems, such as air traffic control systems, may have a lifetime of 30 years or more. Business systems are often more than 10 years old.
- Software cost a lot of money so a company has to use a software system for many years to get a return on its investment.

Types of Changes

- Repair of **Software Faults**-Changing a system to correct deficiencies in the way meets its requirements.
- 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.
- Add to or **modify the system's functionality**-Modifying the system to satisfy new requirements.
- **Improve the program structure and system performance**-Modifying the system without changing functional behavior

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.
- These are sensible observations rather than laws. They are applicable to large systems developed by large organizations.
- Perhaps less applicable in other cases.

Lehman's laws

Law	Description
Continuing change	A program that is used in a real-world environment must necessarily change, or else 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.
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.
Continuing growth	The functionality offered by systems has to continually increase to maintain user satisfaction.
Declining quality	The quality of systems will decline unless they are modified to reflect changes in their operational environment.
Feedback system	Evolution processes incorporate multiagent, multiloop feedback systems and you have to treat them as feedback systems to achieve significant product improvement.

Software Evolution

Broad definition of evolution:

- Generally, software evolution refers to the study and management of the process of making changes to software over time.
- In this definition, software evolution comprises:

❖Development activities

❖Maintenance activities

❖Reengineering activities

Software Maintenance: Overview

The activities of changing the system after it has been delivered

Types of software maintenance:

Corrective maintenance: repair of software faults

Adaptive maintenance: modification of software due to changes in the operating environment (hardware, supporting software)

Perfective maintenance: additions to and/or modifications of system functionality due to organizational or business changes

Development/Maintenance Cost

Maintenance 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.

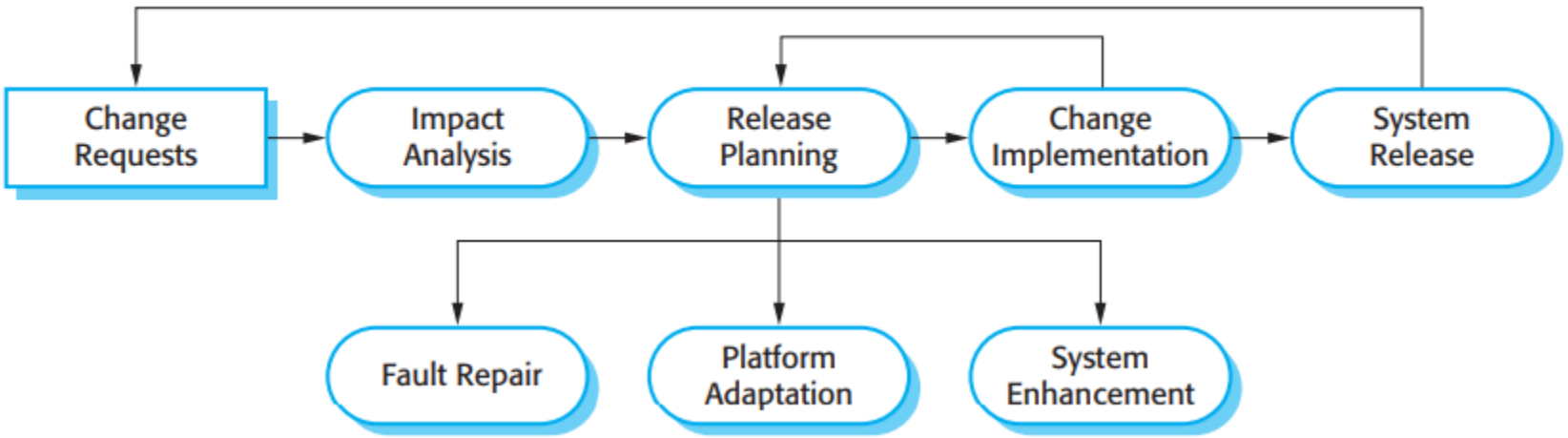
- **Staff skills**

Maintenance staff are often inexperienced and have limited domain knowledge

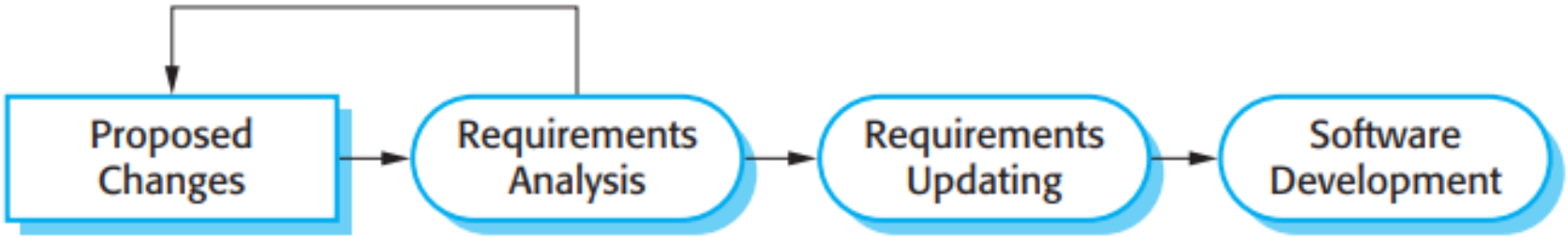
- **Program age and structure**

As programs age, their structure is degraded and they become harder to understand and change.

Software Evolution Process



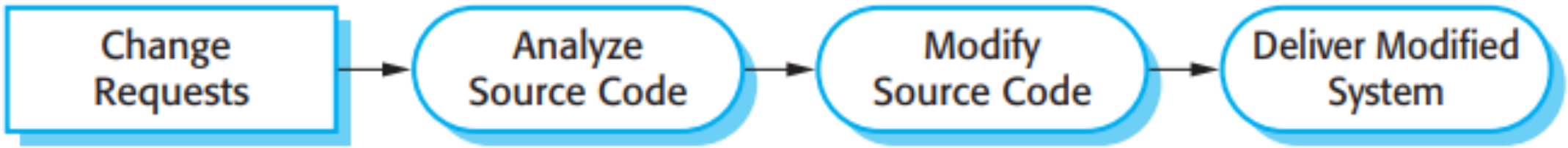
Change Implementation



Urgent Changes Request

- Urgent changes may have to be implemented without going through all stages of the evolution 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).

Emergency Repair Process



Legacy System

- Older Systems that remain vital to an organization
- Legacy systems are old systems that are still useful and are sometimes critical to business operation.
- They may be implemented using outdated languages and technology or may use other systems that are expensive to maintain.
- Often their structure has been degraded by change and documentation is missing or out of date.
- Nevertheless, it may not be cost effective to replace these systems.

The Legacy Dilemma

- It is expensive and risky to replace the legacy system.
- It is expensive to maintain the legacy system.
- Businesses must weigh up the costs and risks and may choose to extend the system lifetime using techniques such as re-engineering.

Software Re-engineering

- It is a process of software development which is done to improve the maintainability of a software system.
- Technical Definition: Software Re-engineering is the examination and alteration of a system to reconstitute it in a new form.