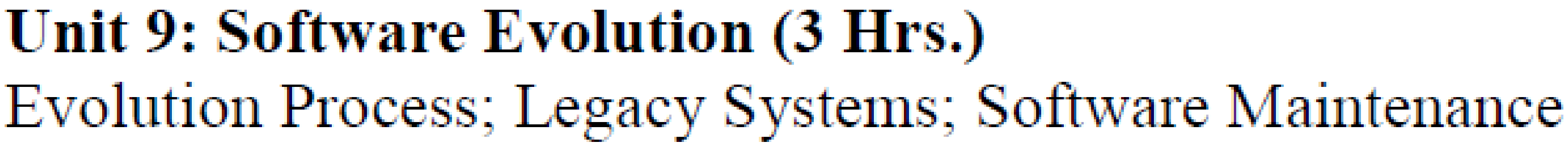
Software Evolution

Review {Testing}

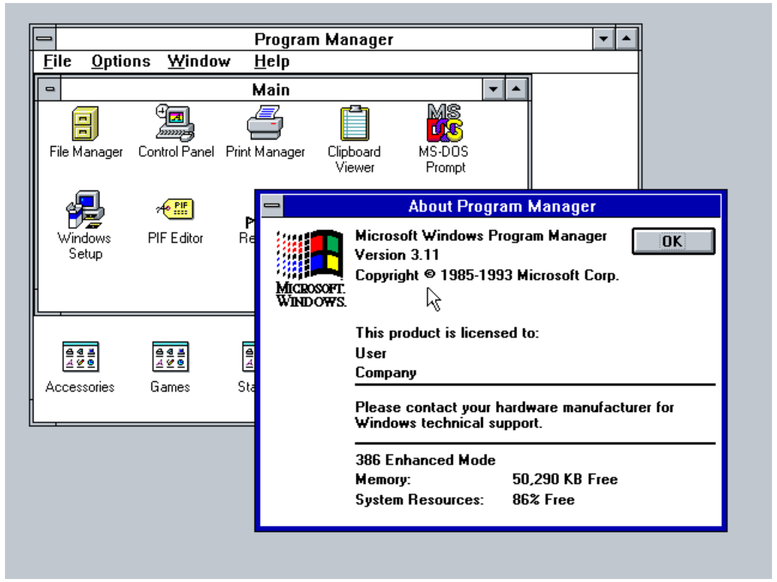
- ✧ Validation & Verification
- ✧ Inspection & Testing
- ✧ Stages of Testing
  - ▪ Development Testing: Unit, Component and System Testing
  - ▪ Release Testing: Requirements testing, Performance Testing
  - ▪ User Testing: Alpha, Beta and Acceptance Testing
- ✧ Test Driven Development



Unit 9: Software Evolution (3 Hrs.)

Evolution Process; Legacy Systems; Software Maintenance

Evolution



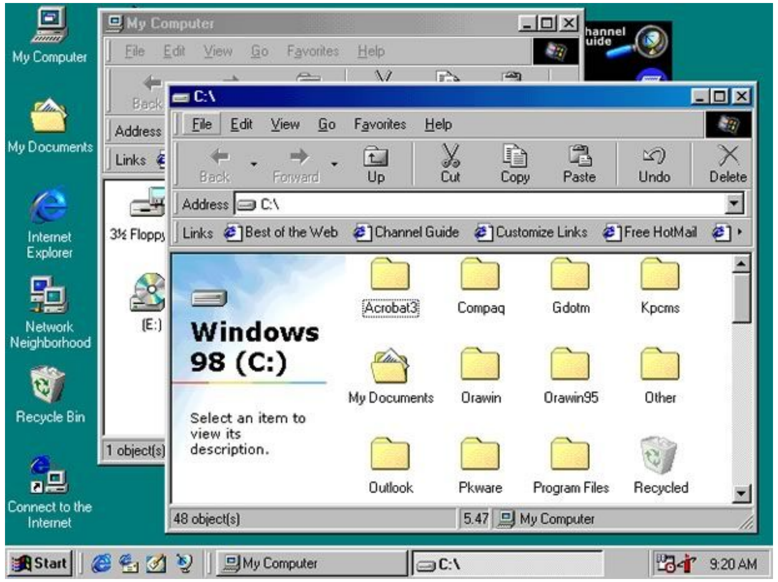
Microsoft Windows Program Manager
Version 3.11
Copyright © 1985-1993 Microsoft Corp.

This product is licensed to:
User
Company

Please contact your hardware manufacturer for Windows technical support.

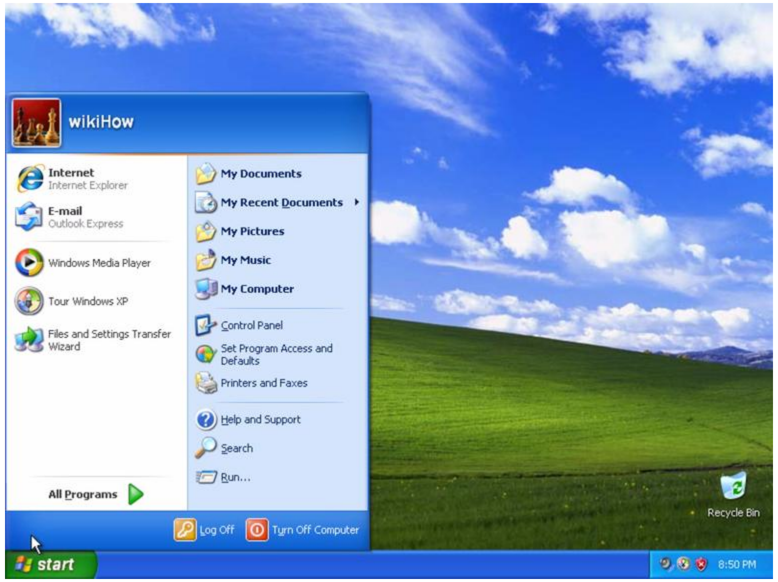
386 Enhanced Mode
Memory: 50,290 KB Free
System Resources: 86% Free

Windows 3.11

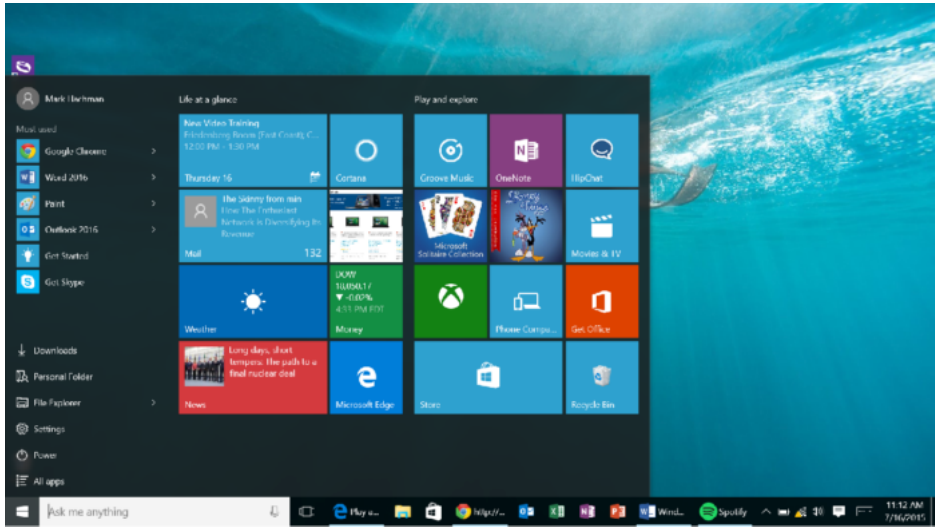

My Computer
My Documents
Internet Explorer
Network Neighborhood
Recycle Bin
Connect to the Internet
C:\
File Edit View Go Favorites Help
Back Forward Up Cut Copy Paste Undo Delete
Address C:\
Links Best of the Web Channel Guide Customize Links Free HotMail
Windows 98 (C:)
Acrobat3 Compaq Gdotm Kpcms
My Documents Orawin Orawin95 Other
Outlook Pkware Program Files Recycled
48 object(s)
5.47 My Computer
Start
9:20 AM

Select an item to view its description.

Windows 98



Windows XP



Windows 10

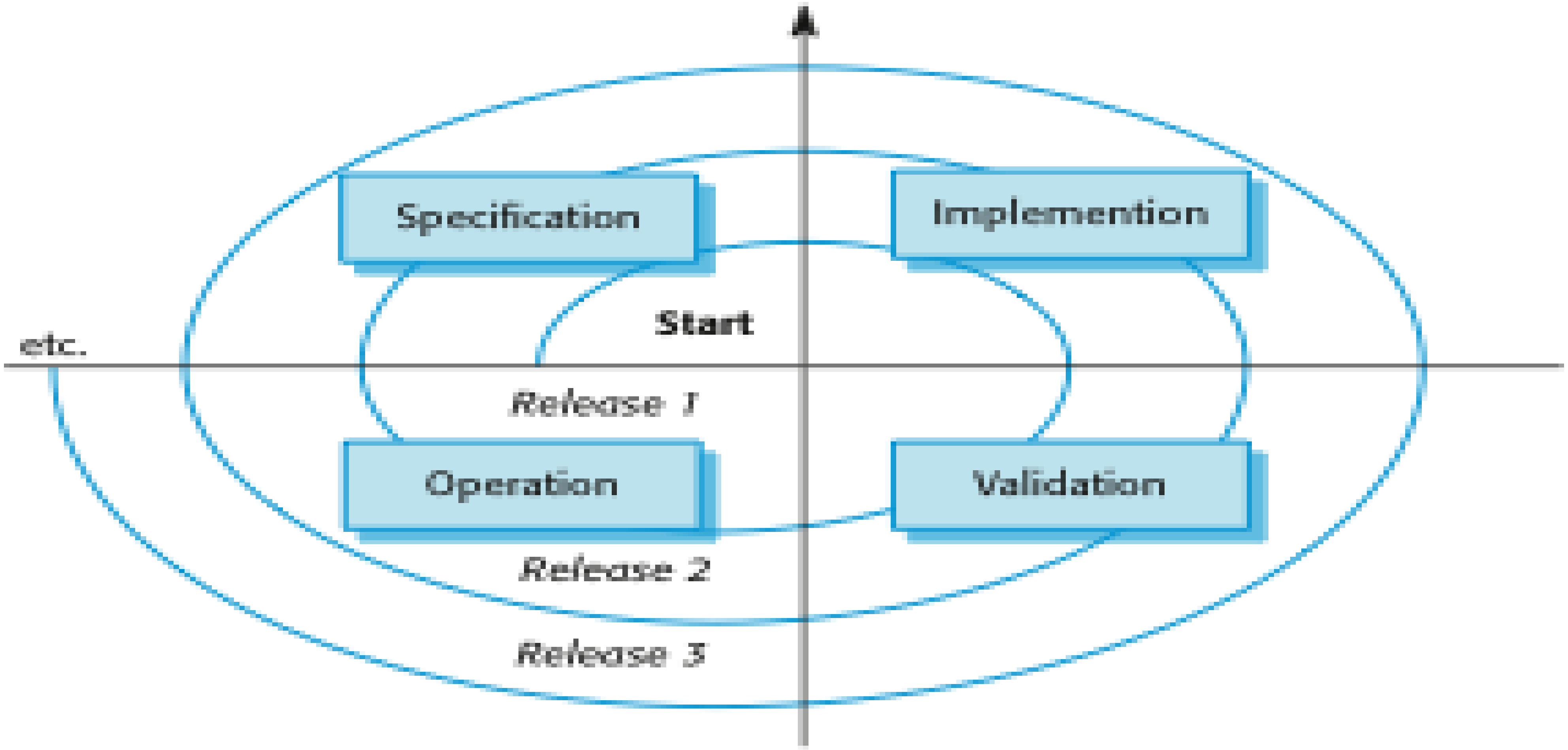
Software change

- ✧ Software change is inevitable
  - New requirements emerge when the software is used;
  - 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.
- ✧ Systems are tightly coupled with their environment. When a system is installed in an environment it changes that environment and therefore changes the system requirements.
- ✧ Systems MUST be changed if they are to remain useful in an environment.
- ✧ A key problem for all organizations is implementing and managing change to their existing software systems.

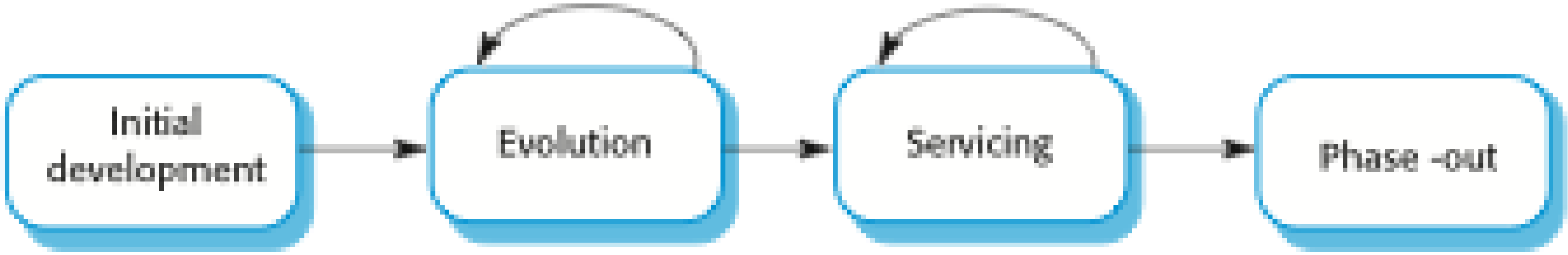
Importance of evolution

- Organizations have hu

A spiral model of development and evolution



Stages of Software



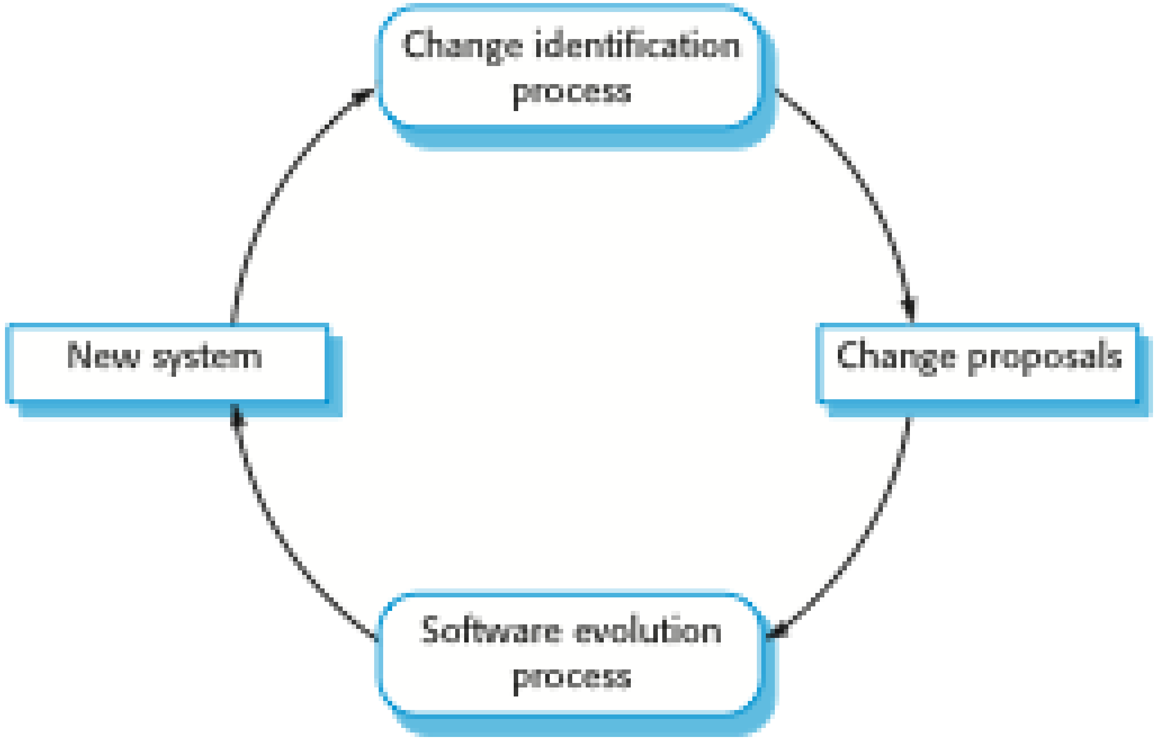
Stages of Software

- 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.

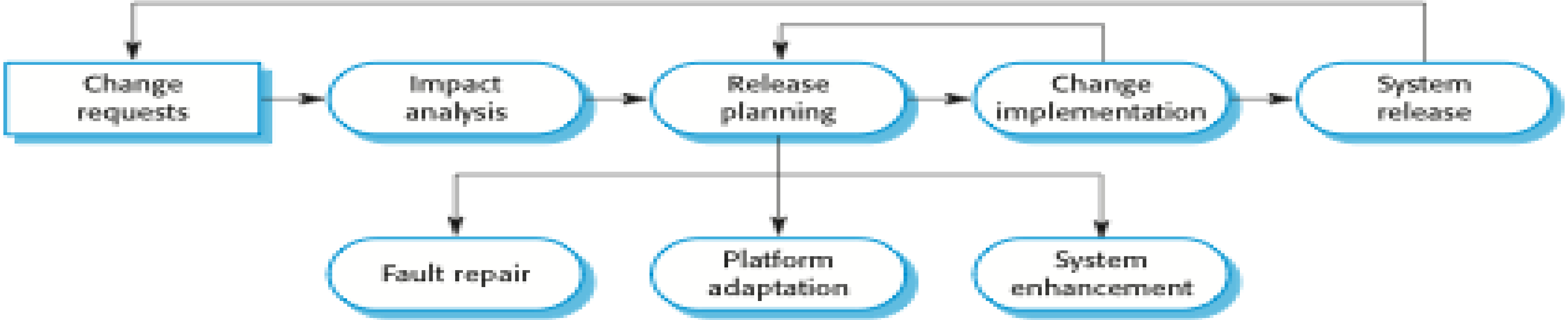
✧ Software evolution

Evolution processes

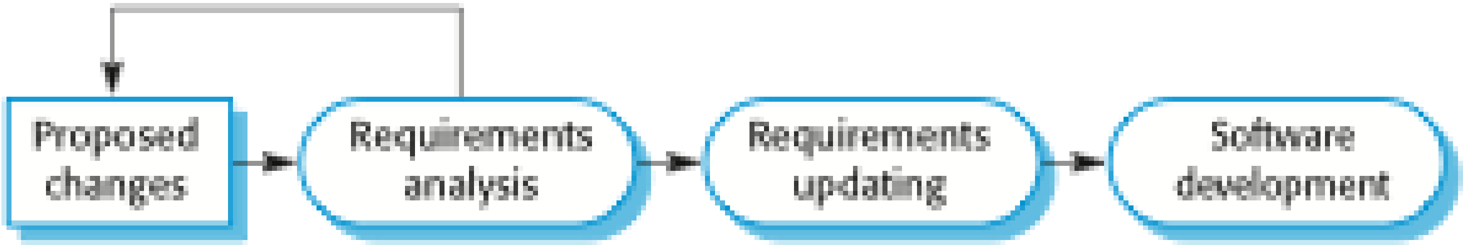
Change identification and evolution processes



The software evolution process



Change implementation



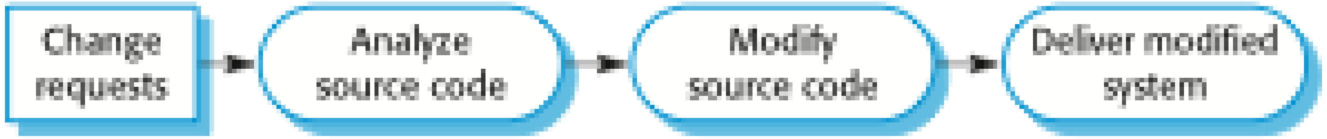
Change implementation

- Iteration of the development process where the revisions to the system are designed, implemented and tested.
- The first stage of change implementation may involve program understanding, especially if the original system developers are not responsible for the change implementation.
- During the program understanding phase, you have to understand how the program is structured, how it delivers functionality and how the proposed change might affect the program.

Urgent change requests

- Urgent changes may

The emergency repair process



Agile methods and evolution

- Agile methods are based on incremental development so the transition from development to evolution is a seamless one.
  - Evolution is simply a continuation of the development process based on frequent system releases.
- Changes may be expressed as additional user stories.

Program evolution dynamics

✧ *Program evolution dyn*

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.

Lehman’s laws

Law	Description
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 multi-agent, multi-loop feedback systems and you have to treat them as feedback systems to achieve significant product improvement.

Software maintenance

- Modifying a program

Types of maintenance

- Maintenance to repa

Maintenance costs

- Usually greater tha

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.

System re-engineering

- Re-structuring or r

Advantages of reengineering

- Reduced risk
  - There is

Reengineering process activities

- Source code transla

Reengineering cost factors

- The quality of the so

Preventative maintenance by refactoring

- ✧ Refactoring is the process of making improvements to a program to slow down degradation through change.
- ✧ You can think of refactoring as ‘preventative maintenance’ that reduces the problems of future change.
- ✧ Refactoring involves modifying a program to improve its structure, reduce its complexity or make it easier to understand.
- ✧ When you refactor a program, you should not add functionality but rather concentrate on program improvement.

Refactoring and reengineering

- ✧ Re-engineering takes place after a system has been maintained for some time and maintenance costs are increasing.
- ✧ Refactoring is a continuous process of improvement throughout the development and evolution process. It is intended to avoid the structure and code degradation that increases the costs and difficulties of maintaining a system.

Legacy system management

✧ Organisations that

Legacy system categories

- Low quality, low bu

✧ Assessment should t

Business value assessment

Issues in business value assessment

- ✧ The use of the system
  - ▪ If systems are only used occasionally or by a small number of people, they may have a low business value.
- ✧ The business processes that are supported
  - ▪ A system may have a low business value if it forces the use of inefficient business processes.
- ✧ System dependability
  - ▪ If a system is not dependable and the problems directly affect business customers, the system has a low business value.
- ✧ The system outputs
  - ▪ If the business depends on system outputs, then the system has a high business value.

System quality assessment

- Business process as