Software maintenance

- Managing the processes of system change

Objectives

- To discuss different types of software maintenance and the maintenance process
- To explain the dynamics of program evolution
- To suggest a number of technical and non-technical factors which affect maintenance costs
- To explain how some complexity metrics may be used to predict maintenance requirements

Topics covered

- The maintenance process
- System documentation
- Program evolution dynamics
- Maintenance costs
- Maintainability measurement

Software maintenance

- Modifying a program after it has been put into use
- Maintenance management is concerned with planning and predicting the process of change
- Configuration management is the management of products undergoing change. Covered in the following chapter

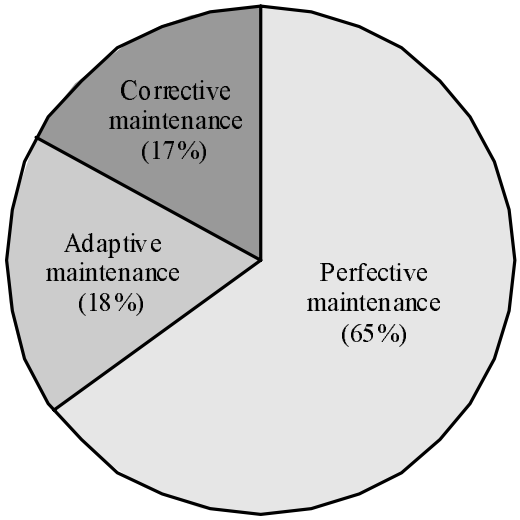
Maintenance is inevitable

- The system requirements are likely to change while the system is being developed because the environment is changing. Therefore a delivered system won't meet its requirements!
- 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 maintained therefore if they are to remain useful in an environment

Types of maintenance

- Corrective maintenance
  - Changing a system to correct deficiencies in the way meets its requirements
- Adaptive maintenance
  - Changing a system to meet new requirements
- Perfective maintenance
  - Changing a system to make it meet its requirements more effectively

Distribution of maintenance effort



Evolving systems

- It is usually more expensive to add functionality after a system has been developed rather than design this into the system
  - Maintenance staff are often inexperienced and unfamiliar with the application domain
  - Programs may be poorly structured and hard to understand
  - Changes may introduce new faults as the complexity of the system makes impact assessment difficult
  - The structure may be degraded due to continual change
  - There may be no documentation available to describe the program

Maintenance management

- Maintenance has a poor image amongst development staff as it is not seen as challenging and creative
- Maintenance costs increase as the software is maintained
- The amount of software which has to be maintained increases with time
- Inadequate configuration management often means that the different representations of a system are out of step

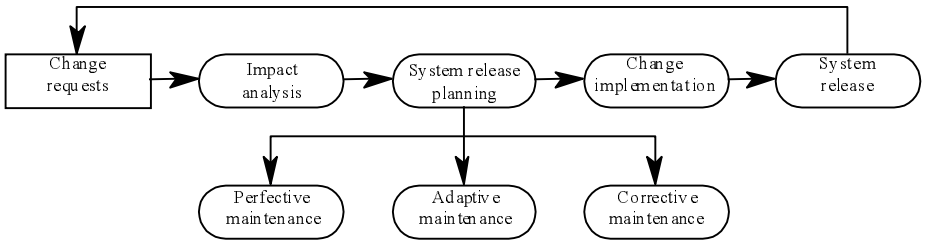
Staff motivation

- Relate software development to organizational goals - maintenance rationale
- Relate rewards to organizational performance
- Integrate maintenance with development
- Create a discretionary preventative maintenance budget
- Plan for maintenance early in the development process
- Plan to expend effort on program maintainability

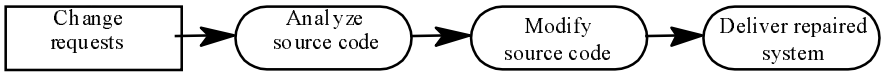
The maintenance process

- Maintenance is triggered by change requests from customers or marketing requirements
- Changes are normally batched and implemented in a new release of the system
- Programs sometimes need to be repaired without a complete process iteration but this is dangerous as it leads to documentation and programs getting out of step

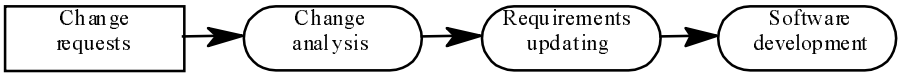
The maintenance process



Change processes



Fault repair process



Iterative development process

System documentation

- Requirements document
- System architecture description
- Program design documentation
- Source code listings
- Test plans and validation reports
- System maintenance guide

Document production

- Structure documents with overviews leading the reader into more detailed technical descriptions
- Produce good quality, readable manuals - they may have to last 20 years
- Use tool-generated documentation whenever possible

Program evolution dynamics

- Program evolution dynamics is the study of the processes of system change
- After major empirical study, 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

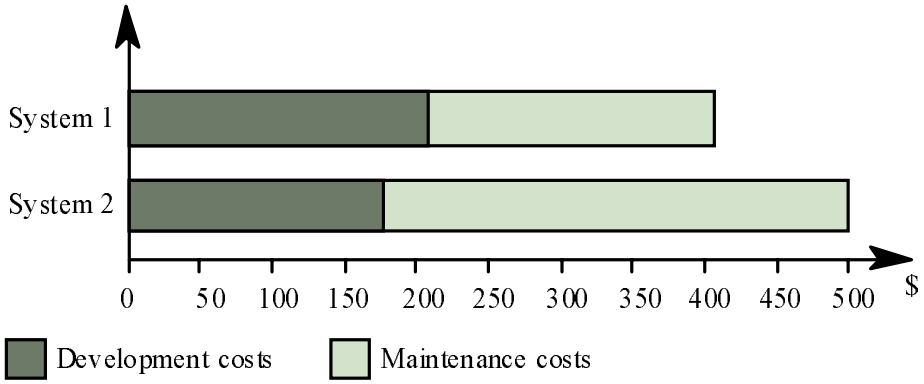
Lehman's laws

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 are 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.

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.)

Development/maintenance costs



Maintenance cost factors

- Module independence
  - It should be possible to change one module without affecting others
- Programming language
  - High-level language programs are easier to maintain
- Programming style
  - Well-structured programs are easier to maintain
- Program validation and testing
  - Well-validated programs tend to require fewer changes due to corrective maintenance

Maintenance cost factors

- Documentation
  - Good documentation makes programs easier to understand
- Configuration management
  - Good CM means that links between programs and their documentation are maintained
- Application domain
  - Maintenance is easier in mature and well-understood application domains
- Staff stability
  - Maintenance costs are reduced if the same staff are involved with them for some time

Maintenance cost factors

- Program age
  - The older the program, the more expensive it is to maintain (usually)
- External environment
  - If a program is dependent on its external environment, it may have to be changed to reflect environmental changes
- Hardware stability
  - Programs designed for stable hardware will not require to change as the hardware changes

Maintenance cost estimation

- Can use COCOMO model for maintenance cost estimation
- Based on % of program instructions changed per year (ACT)
- Simplistically, maintenance costs are directly proportional to development costs
- Multipliers like development multipliers but with different values

Maintenance planning

- COCOMO approach is simplistic because of non-technical factors
- COCOMO approach can be used to decide what and what kind of resources should be devoted to maintenance

Maintenance/development effort

- Replace with portrait slide

Maintenance metrics

- Measurements of program characteristics which would allow maintainability to be predicted
- Essentially technical, how can technical factors above be quantified
- Any software components whose measurements are out of line with other components may be excessively expensive to maintain. Perhaps perfective maintenance effort should be devoted to these components

Maintenance metrics

- *Control complexity* Can be measured by examining the conditional statements in the program
- *Data complexity* Complexity of data structures and component interfaces.
- *Length of identifier names* Longer names imply readability
- *Program comments* Perhaps more comments mean easier maintenance

Maintenance metrics

- *Coupling* How much use is made of other components or data structures
- *Degree of user interaction* The more user I/O, the more likely the component is to require change
- *Speed and space requirements* Require tricky programming, harder to maintain

Process 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

Maintenance metrics

- Log maintenance effort on a per component basis
- Choose set of possible metrics which may be related to maintenance
- Assess possible metrics for each maintained components
- Look for correlation between maintenance effort and metric values

Key points

- Three types of maintenance are perfective, adaptive and corrective
- Maintenance costs usually exceed development costs for large, long-lifetime systems
- Investing effort in maintainability is therefore likely to be cost-effective in the long-term
- Documentation should include requirements, design and validation documents

Key points

- Lehman's laws of program evolution dynamics have been derived from empirical observation
- Technical and non-technical factors affect maintenance costs
- Complexity metrics may be useful in finding components which cause maintenance problems