Software change

- Software change is inevitable
  - New requirements emerge when the software is used
  - The business environment changes
  - Errors must be repaired
  - New equipment must be accommodated
  - The performance or reliability may have to be improved
- A key problem for organizations is implementing and managing change to their legacy systems

Discussion

- Project
  - A change to your project
  - A change to someone else’s project
  - Another team in your class
  - Another team from 2000
  - Another team from 1990

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

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 if they are to remain useful in an environment

Types of maintenance

- Maintenance to repair software faults
  - Changing a system to correct deficiencies in the way it 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
- Which one is most common?

Distribution of maintenance effort

- Fault repair (17%)
- Software adaptation (18%)
- Functionality addition or modification (65%)

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 future maintenance more difficult.
- Ageing software can have high support costs (e.g. old languages, compilers etc.)
Development/maintenance costs

- Team stability
  - Maintenance costs are reduced if the same staff is 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 is often inexperienced and has limited domain knowledge
- Program age and structure
  - As programs age, their structure is degraded and they become harder to understand and change

Change requests
- Change requests are requests for system changes from users, customers or management
- In principle, all change requests should be carefully analyzed as part of the maintenance process and then implemented
- In practice, some change requests must be implemented urgently
  - Fault repair
  - Changes to the system’s environment
  - Urgently required business changes

The maintenance process

Maintenance prediction
- Maintenance prediction is concerned with assessing what 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
Change prediction

- Predicting the number of changes requires understanding 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
  - The business processes where the system is used

Evolutionary software

- Rather than think of separate development and maintenance phases, evolutionary software is software that is designed so that it can continuously evolve throughout its lifetime

Configuration management

- New versions of software systems are created as they change
  - For different machines/OS
  - Offering different functionality
  - Tailored for particular user requirements
- Configuration management is concerned with managing evolving software systems
  - System change is a team activity
  - CM aims to control the costs and effort involved in making changes to a system

Configuration management planning

- All products of the software process may have to be managed
  - Specifications
  - Designs
  - Programs
  - Test data
  - User manuals
- Thousands of separate documents are generated for a large software system
- DISCUSSION
The CM plan

• Defines the types of documents to be managed and a document naming scheme
• Defines who takes responsibility for the CM procedures
• Defines policies for change and version management
• Defines the CM records which must be maintained

The CM plan

• Describes the tools which should be used to assist the CM process and any limitations on their use
• Defines the process of tool use
• Defines the CM database used to record configuration information
• May include information such as the CM of external software, process auditing, etc.

The configuration database

• All CM information should be maintained in a configuration database
• This should allow queries about configurations to be answered
  - Who has a particular system version?
  - What platform is required for a particular version?
  - What versions are affected by a change to component X?
  - How many reported faults in version T?
• The CM database should preferably be linked to the software being managed