Software Maintenance

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

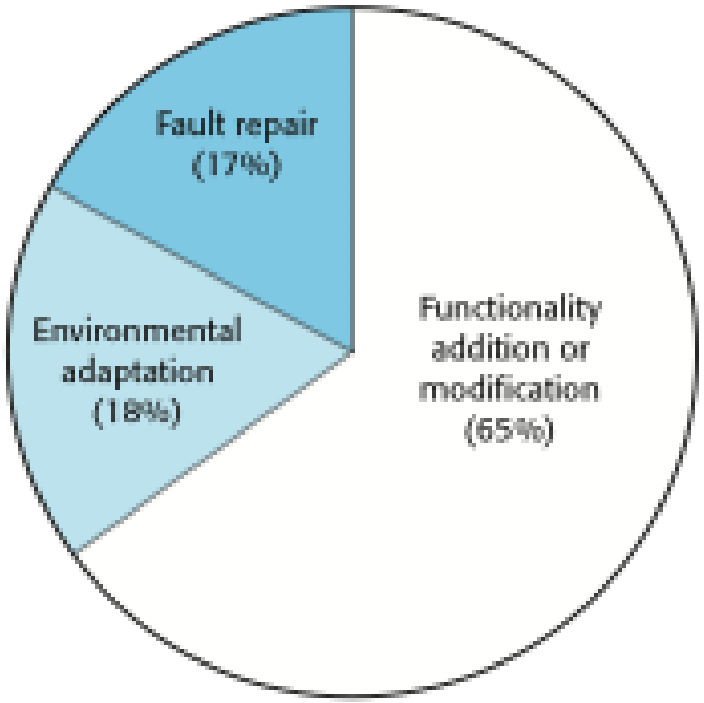
Software Maintenance

- Modifying a program after it has been put into use.
- The term is mostly used for changing custom software. Generic software products are said to evolve to create new versions.
- 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

- Corrective maintenance
  - Reactive modification of a software product performed after delivery to correct discovered problems
- Adaptive maintenance
  - Modification of a software product performed after delivery to keep a software product usable in a changed or changing environment
- Perfective maintenance
  - Modification of a software product after delivery to improve performance or maintainability
- Preventive maintenance
  - Modification of a software product after delivery to detect and correct latent faults in the software product before they become effective faults

Maintenance effort distribution



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.).

Characteristics of SW Maintenance – Lehman's Law

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.

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 of reengineering
  - 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.

Reengineering Process Activities

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

Configuration Management

- Because software changes frequently, systems can be thought of as a set of versions, each of which has to be maintained and managed.
- Versions implement proposals for change, corrections of faults, and adaptations for different hardware and operating systems.
- Configuration management (CM) is concerned with the policies, processes and tools for managing changing software systems. You need CM because it is easy to lose track of what changes and component versions have been incorporated into each system version.

Configuration Management Activities

- Change management
  - Keeping track of requests for changes to the software from customers and developers, working out the costs and impact of changes, and deciding the changes should be implemented.
- Version management
  - Keeping track of the multiple versions of system components and ensuring that changes made to components by different developers do not interfere with each other.
- System building
  - The process of assembling program components, data and libraries, then compiling these to create an executable system.
- Release management
  - Preparing software for external release and keeping track of the system versions that have been released for customer use.

Change Management

- Organizational needs and requirements change during the lifetime of a system, bugs have to be repaired and systems have to adapt to changes in their environment.
- Change management is intended to ensure that system evolution is a managed process and that priority is given to the most urgent and cost-effective changes.
- The change management process is concerned with analyzing the costs and benefits of proposed changes, approving those changes that are worthwhile and tracking which components in the system have been changed.

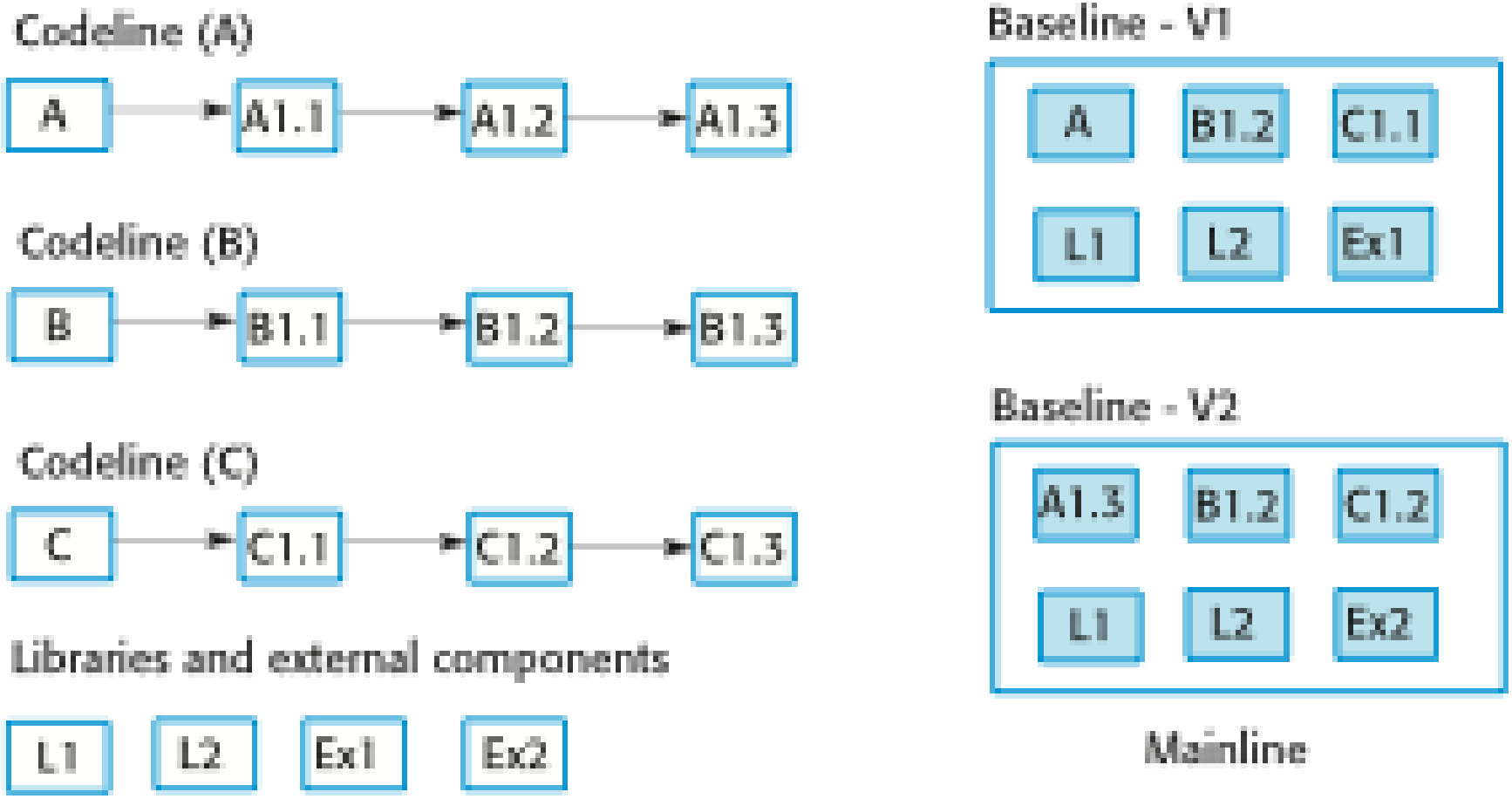
Version Management

- Version management (VM) is the process of keeping track of different versions of software components or configuration items and the systems in which these components are used.
- It also involves ensuring that changes made by different developers to these versions do not interfere with each other.
- Therefore version management can be thought of as the process of managing codelines and baselines.

Codelines and Baselines

- A codeline is a sequence of versions of source code with later versions in the sequence derived from earlier versions.
- Codelines normally apply to components of systems so that there are different versions of each component.
- A baseline is a definition of a specific system.
- The baseline therefore specifies the component versions that are included in the system plus a specification of the libraries used, configuration files, etc.

Codelines and Baselines



System Building

- System building is the process of creating a complete, executable system by compiling and linking the system components, external libraries, configuration files, etc.
- System building tools and version management tools must communicate as the build process involves checking out component versions from the repository managed by the version management system.
- The configuration description used to identify a baseline is also used by the system building tool.

Release Management

- A system release is a version of a software system that is distributed to customers.
- For mass market software, it is usually possible to identify two types of release: major releases which deliver significant new functionality, and minor releases, which repair bugs and fix customer problems that have been reported.
- For custom software or software product lines, releases of the system may have to be produced for each customer and individual customers may be running several different releases of the system at the same time.