Software maintenance in software engineering is the modification of a software product after delivery to correct faults, to improve performance or other attributes.

A common perception of maintenance is that it merely involves fixing defects. However, one study indicated that over 80% of maintenance effort is used for non-corrective actions. This perception is perpetuated by users submitting problem reports that in reality are functionality enhancements to the system. More recent studies put the bug-fixing proportion closer to 21%.

Software maintenance and evolution of systems was first addressed by Meir M. Lehman in 1969. Over a period of twenty years, his research led to the formulation of Lehman's Laws (Lehman 1997). Key findings of his research conclude that maintenance is really evolutionary development and that maintenance decisions are aided by understanding what happens to systems (and software) over time. Lehman demonstrated that systems continue to evolve over time. As they evolve, they grow more complex unless some action such as code refactoring is taken to reduce the complexity.

In the late 1970s, a famous and widely cited survey study by Lientz and Swanson, exposed the very high fraction of life-cycle costs that were being expended on maintenance. They categorized maintenance activities into four classes:

- Adaptive – modifying the system to cope with changes in the software environment (DBMS, OS)
- Perfective – implementing new or changed user requirements which concern functional enhancements to the software
- Corrective – diagnosing and fixing errors, possibly ones found by users
- Preventive – increasing software maintainability or reliability to prevent problems in the future

The survey showed that around 75% of the maintenance effort was on the first two types, and error correction consumed about 21%. Many subsequent studies suggest a similar problem magnitude. Studies show that contribution of end users is crucial during the new requirement data gathering and analysis. This is the main cause of any problem during software evolution and maintenance. Software maintenance is important because it consumes a large part of the overall lifecycle costs and also the inability to change software quickly and reliably means that business opportunities are lost.

Importance of software maintenance

The key software maintenance issues are both managerial and technical. Key management issues are: alignment with customer priorities, staffing, which organization does maintenance, estimating costs. Key technical issues are: limited understanding, impact analysis, testing, maintainability measurement.

Software maintenance is a very broad activity that includes error correction, enhancements of capabilities, deletion of obsolete capabilities, and optimization. Because change is inevitable, mechanisms must be developed for evaluation, controlling and making modifications.

So any work done to change the software after it is in operation is considered to be maintenance work. The purpose is to preserve the value of software over the time. The value can be enhanced by expanding the customer base, meeting additional requirements, becoming easier to use, more efficient and employing newer technology. Maintenance may span for 20 years, whereas development may be 1–2 years.

Software maintenance planning

An integral part of software is the maintenance one, which requires an accurate maintenance plan to be prepared during the software development. It should specify how users will request modifications or report problems. The budget should include resource and cost estimates. A new decision should be addressed for the developing of every new system feature and its quality objectives. The software

maintenance, which can last for 5–6 years (or even decades) after the development process, calls for an effective plan which can address the scope of software maintenance, the tailoring of the post delivery/deployment process, the designation of who will provide maintenance, and an estimate of the life-cycle costs. The selection of proper enforcement of standards is the challenging task right from early stage of software engineering which has not got definite importance by the concerned stakeholders.

Software maintenance processes

This section describes the six software maintenance processes as:

1. The implementation process contains software preparation and transition activities, such as the conception and creation of the maintenance plan; the preparation for handling problems identified during development; and the follow-up on product configuration management.
2. The problem and modification analysis process, which is executed once the application has become the responsibility of the maintenance group. The maintenance programmer must analyze each request, confirm it (by reproducing the situation) and check its validity, investigate it and propose a solution, document the request and the solution proposal, and finally, obtain all the required authorizations to apply the modifications.
3. The process considering the implementation of the modification itself.
4. The process acceptance of the modification, by confirming the modified work with the individual who submitted the request in order to make sure the modification provided a solution.
5. The migration process (platform migration, for example) is exceptional, and is not part of daily maintenance tasks. If the software must be ported to another platform without any change in functionality, this process will be used and a maintenance project team is likely to be assigned to this task.
6. Finally, the last maintenance process, also an event which does not occur on a daily basis, is the retirement of a piece of software.

There are a number of processes, activities and practices that are unique to maintainers, for example:

- Transition: a controlled and coordinated sequence of activities during which a system is transferred progressively from the developer to the maintainer;
- Service Level Agreements (SLAs) and specialized (domain-specific) maintenance contracts negotiated by maintainers;
- Modification Request and Problem Report Help Desk: a problem-handling process used by maintainers to prioritize, documents and route the requests they receive;

Categories of maintenance

These have since been updated and ISO/IEC 14764 presents:

- 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.

There is also a notion of pre-delivery/pre-release maintenance which is all the good things you do to lower the total cost of ownership of the software. Things like compliance with coding standards that includes software maintainability goals. The management of coupling and cohesion of the software. The attainment of software supportability goals (SAE JA1004, JA1005 and JA1006 for example). Note also

that some academic institutions are carrying out research to quantify the cost to ongoing software maintenance due to the lack of resources such as design documents and system/software comprehension training and resources (multiply costs by approx. 1.5-2.0 where there is no design data available).

Configuration management (**CM**) is a systems engineering process for establishing and maintaining consistency of a product's performance, functional, and physical attributes with its requirements, design, and operational information throughout its life.[1][2] The CM process is widely used by military engineering organizations to manage changes throughout the system lifecycle of complex systems, such as weapon systems, military vehicles, and information systems. Outside the military, the CM process is also used with IT service management as defined by ITIL, and with other domain models in the civil engineering and other industrial engineering segments such as roads, bridges, canals, dams, and buildings.

CM applied over the life cycle of a system provides visibility and control of its performance, functional, and physical attributes. CM verifies that a system performs as intended, and is identified and documented in sufficient detail to support its projected life cycle. The CM process facilitates orderly management of system information and system changes for such beneficial purposes as to revise capability; improve performance, reliability, or maintainability; extend life; reduce cost; reduce risk and liability; or correct defects. The relatively minimal cost of implementing CM is returned many fold in cost avoidance. The lack of CM, or its ineffectual implementation, can be very expensive and sometimes can have such catastrophic consequences such as failure of equipment or loss of life.

CM emphasizes the functional relation between parts, subsystems, and systems for effectively controlling system change. It helps to verify that proposed changes are systematically considered to minimize adverse effects. Changes to the system are proposed, evaluated, and implemented using a standardized, systematic approach that ensures consistency, and proposed changes are evaluated in terms of their anticipated impact on the entire system. CM verifies that changes are carried out as prescribed and that documentation of items and systems reflects their true configuration. A complete CM program includes provisions for the storing, tracking, and updating of all system information on a component, subsystem, and system basis.

A structured CM program ensures that documentation (e.g., requirements, design, test, and acceptance documentation) for items is accurate and consistent with the actual physical design of the item. In many cases, without CM, the documentation exists but is not consistent with the item itself. For this reason, engineers, contractors, and management are frequently forced to develop documentation reflecting the actual status of the item before they can proceed with a change. This reverse engineering process is wasteful in terms of human and other resources and can be minimized or eliminated using CM.

Overview

CM is the practice of handling changes systematically so that a system maintains its integrity over time. CM implements the policies, procedures, techniques, and tools that manage, evaluate proposed changes, track the status of changes, and maintain an inventory of system and support documents as the system changes. CM programs and plans provide technical and administrative direction to the development and implementation of the procedures, functions, services, tools, processes, and resources required to successfully develop and support a complex system. During system development, CM allows program management to track requirements throughout the life-cycle through acceptance and operations and maintenance. As changes inevitably occur in the requirements and design, they must be approved and documented, creating an accurate record of the system status. Ideally the CM process is applied throughout the system lifecycle. Most professionals mix up or get confused with Asset management (AM), where it inventories the assets on hand. The key difference between CM and AM is that the former does not manage the financial accounting aspect but on service that the system supports.

The CM process for both hardware- and software-configuration items comprises five distinct disciplines as established in the MIL–HDBK–61A[9] and in ANSI/EIA-649. These disciplines are carried out[*by whom?*] as policies and procedures for establishing baselines and for performing a standard change-

management process. The IEEE 12207 process IEEE 12207.2 also has these activities and adds "Release management and delivery".

The five disciplines are:

1. CM Planning and Management: a formal document and plan to guide the CM program that includes items such as:
    - personnel
    - responsibilities and resources
    - training requirements
    - administrative meeting guidelines, including a definition of procedures and tools
    - baselining processes
    - configuration control and configuration-status accounting
    - naming conventions
    - audits and reviews
    - subcontractor/vendor CM requirements
2. Configuration Identification (CI): consists of setting and maintaining baselines, which define the system or subsystem architecture, components, and any developments at any point in time. It is the basis by which changes to any part of a system are identified, documented, and later tracked through design, development, testing, and final delivery. CI incrementally establishes and maintains the definitive current basis for Configuration Status Accounting (CSA) of a system and its configuration items (CIs) throughout their lifecycle (development, production, deployment, and operational support) until disposal.
3. Configuration Control: includes the evaluation of all change-requests and change-proposals, and their subsequent approval or disapproval. It covers the process of controlling modifications to the system's design, hardware, firmware, software, and documentation.
4. Configuration Status Accounting: includes the process of recording and reporting configuration item descriptions (e.g., hardware, software, firmware, etc.) and all departures from the baseline during design and production. In the event of suspected problems, the verification of baseline configuration and approved modifications can be quickly determined.
5. Configuration Verification and Audit: an independent review of hardware and software for the purpose of assessing compliance with established performance requirements, commercial and appropriate military standards, and functional, allocated, and product baselines. Configuration audits verify that the system and subsystem configuration documentation complies with the functional and physical performance characteristics before acceptance into an architectural baseline.

Software configuration management (SCM)

The software configuration management (SCM) process is looked upon by practitioners as the best solution to handling changes in software projects. It identifies the functional and physical attributes of software at various points in time, and performs systematic control of changes to the identified attributes for the purpose of maintaining software integrity and traceability throughout the software development life cycle.

The SCM process further defines the need to trace changes, and the ability to verify that the final delivered software has all of the planned enhancements that are supposed to be included in the release. It identifies four procedures that must be defined for each software project to ensure that a sound SCM process is implemented. They are:

1. Configuration identification
2. Configuration control
3. Configuration status accounting
4. Configuration audits

These terms and definitions change from standard to standard, but are essentially the same.

- Configuration identification is the process of identifying the attributes that define every aspect of a configuration item. A configuration item is a product (hardware and/or software) that has an end-user purpose. These attributes are recorded in configuration documentation and baselined. Baselining an attribute forces formal configuration change control processes to be effected in the event that these attributes are changed.
- Configuration change control is a set of processes and approval stages required to change a configuration item's attributes and to re-baseline them.
- Configuration status accounting is the ability to record and report on the configuration baselines associated with each configuration item at any moment of time.
- Configuration audits are broken into functional and physical configuration audits. They occur either at delivery or at the moment of effecting the change. A functional configuration audit ensures that functional and performance attributes of a configuration item are achieved, while a physical configuration audit ensures that a configuration item is installed in accordance with the requirements of its detailed design documentation.

Configuration management database

The Information Technology Infrastructure Library (ITIL) specifies the use of a Configuration management system (CMS) or Configuration management database (CMDB) as a means of achieving industry best practices for Configuration Management. CMDBs are used to track Configuration Items (CIs) and the dependencies between them, where CIs represent the things in an enterprise that are worth tracking and managing, such as but not limited to computers, software, software licenses, racks, network devices, storage, and even the components within such items.

The benefits of a CMS/CMDB includes being able to perform functions like root cause analysis, impact analysis, change management, and current state assessment for future state strategy development. Example systems, commonly identifies themselves as IT Service Management (ITSM) systems, include FreshService, ServiceNow and Samanage.

Information assurance

For information assurance, CM can be defined as the management of security features and assurances through control of changes made to hardware, software, firmware, documentation, test, test fixtures, and test documentation throughout the life cycle of an information system.[10] CM for information assurance, sometimes referred to as **S**ecure **C**onfiguration **M**anagement, relies upon performance, functional, and physical attributes of IT platforms and products and their environments to determine the appropriate security features and assurances that are used to measure a system configuration state. For example, configuration requirements may be different for a network firewall that functions as part of an organization's Internet boundary versus one that functions as an internal local network firewall.

Maintenance systems

Configuration management is used to maintain an understanding of the status of complex assets with a view to maintaining the highest level of serviceability for the lowest cost. Specifically, it aims to ensure that operations are not disrupted due to the asset (or parts of the asset) overrunning limits of planned lifespan or below quality levels.

In the military, this type of activity is often classed as "mission readiness", and seeks to define which assets are available and for which type of mission; a classic example is whether aircraft on board an aircraft carrier are equipped with bombs for ground support or missiles for defense.

Operating System configuration management

Configuration management can be used to maintain OS configuration files.[11] Example systems include Ansible, Bcfg2, CFEngine, Chef, Otter, Puppet, Quattor, SaltStack, and Vagrant. Many of these systems utilize Infrastructure as Code to define and maintain configuration.[12]

A theory of configuration maintenance was worked out by Mark Burgess,[13][14][15] with a practical implementation on present day computer systems in the software CFEngine able to perform real time repair as well as preventive maintenance.

Preventive maintenance

Understanding the "as is" state of an asset and its major components is an essential element in preventive maintenance as used in maintenance, repair, and overhaul and enterprise asset management systems.

Complex assets such as aircraft, ships, industrial machinery etc. depend on many different components being serviceable. This serviceability is often defined in terms of the amount of usage the component has had since it was new, since fitted, since repaired, the amount of use it has had over its life and several other limiting factors. Understanding how near the end of their life each of these components is has been a major undertaking involving labor-intensive record keeping until recent developments in software.

Predictive maintenance

Many types of component use electronic sensors to capture data which provides live condition monitoring. This data is analyzed on board or at a remote location by computer to evaluate its current serviceability and increasingly its likely future state using algorithms which predict potential future failures based on previous examples of failure through field experience and modeling. This is the basis for "predictive maintenance".

Availability of accurate and timely data is essential in order for CM to provide operational value and a lack of this can often be a limiting factor. Capturing and disseminating the operating data to the various support organizations is becoming an industry in itself.

The consumers of this data have grown more numerous and complex with the growth of programs offered by original equipment manufacturers (OEMs). These are designed to offer operators guaranteed availability and make the picture more complex with the operator managing the asset but the OEM taking on the liability to ensure its serviceability.