**Software Maintenance**

Table of Contents

[Types of Maintenance 3](#_Toc83393993)

[Corrective Maintenance 3](#_Toc83393994)

[Adaptive Maintenance 3](#_Toc83393995)

[Perfective Maintenance 3](#_Toc83393996)

[Preventive Maintenance 4](#_Toc83393997)

[Cost of Maintenance 5](#_Toc83393998)

[Maintenance Activities 6](#_Toc83393999)

[Software Re-Engineering 8](#_Toc83394000)

[Reuse 9](#_Toc83394001)

[Reuse Process 9](#_Toc83394002)

**Software Maintenance** consists of more than just fixing mistakes. It is any **post-delivery modifications** to a system. These may be corrective, adaptive, perfective or preventive.

## Types of Maintenance

### Corrective Maintenance

**Corrective Maintenance** includes modifications done to fix problems which are **discovered by users**. The maintenance tasks can be both **planned** and **unplanned**.

There are three situations where corrective maintenance occurs:

1. When an issue is detected through condition monitoring
2. When a routine inspection uncovers a potential fault
3. When a piece of equipment breaks down

### Adaptive Maintenance

**Adaptive Maintenance** keeps software **up-to-date** and tune to the ever changing world. It addresses new knowledge, hardware and security threats. They tend to deal with **internal workings** and thus have low impact on users. In fact, users are more likely to notice if we do not perform adaptive maintenance.

### Perfective Maintenance

**Perfective Maintenance** keeps software **usable** over a long period of time by including new features, new user requirements and improved reliability and performance. As such, these changes are **highly noticeable** and requires communication with users.

### Preventive Maintenance

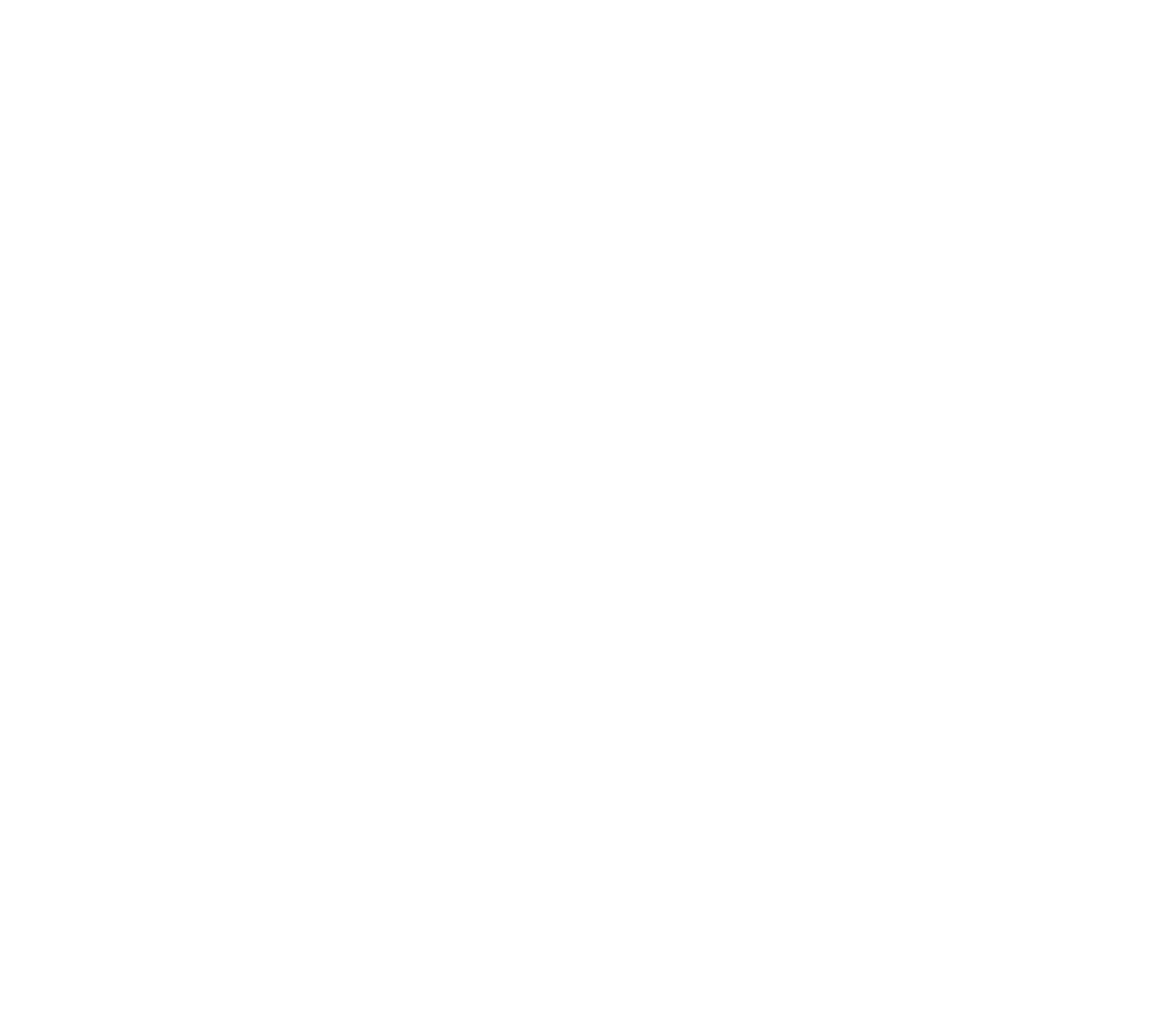
**Preventive Maintenance** stops **future problems** before they become serious. This includes making code easier to scale or maintain and finding and fixing faults that have not caused any problems yet.

## Cost of Maintenance

The cost of the **maintenance stage** is as high as 67% of the total software development cycle’s cost. This is due to a variety of factors:

* Standard age of software is considered up to 10 to 15 yrs.
* Older software, which were meant to work on slow machines with less memory and storage capacity cannot keep themselves challenging against newly coming enhanced software on modern hardware.
* As technology advances, it becomes costly to maintain old SW.
* Most maintenance engineers are newbie and use trial and error method to rectify problem.
* Changes are often left undocumented which may cause more conflicts in future.

## Maintenance Activities



* **Identification & Tracing** - It involves activities pertaining to identification of requirement of modification or maintenance. It is generated by user or system may itself report via logs or error messages.
* **Analysis** - The modification is analyzed for its impact on the system including safety and security implications. If probable impact is severe, alternative solution is looked for. A set of required modifications is then materialized into requirement specifications. The cost of modification/maintenance is analyzed and estimation is concluded.
* **Design** - New modules, which need to be replaced or modified, are designed against requirement specifications set in the previous stage. Test cases are created for validation and verification.
* **Implementation** - The new modules are coded with the help of structured design created in the design step. Every programmer is expected to do unit testing in parallel.
* **System Testing** - Integration testing is done among newly created modules. Integration testing is also carried out between new modules and the system. Finally, the system is tested as a whole, following regressive testing procedures.
* **Acceptance Testing** - After testing the system internally, it is tested for acceptance with the help of users. If at this state, user complaints some issues they are addressed or noted to address in next iteration.
* **Delivery** - After acceptance test, the system is deployed all over the organization either by small update package or fresh installation of the system. The final testing takes place at client end after the software is delivered.
* **Maintenance management** - Configuration management is an essential part of system maintenance. It is aided with version control tools to control versions, semi-version or patch management.

## Software Re-Engineering

When we need to **update software** without impacting its **functionality**, it is called **software re-engineering**. The design of the software is changed and the programs are re-written to keep it in the current market.

* **Decide** what to re-engineer. Is it whole software or a part of it?
* **Perform** Reverse Engineering, in order to obtain specifications of existing software.
* **Restructure Program** if required. For example, changing function-oriented programs into object-oriented programs.
* **Re-structure data** as required.
* **Apply Forward engineering** concepts in order to get re-engineered software.

Here, the **Reverse Engineering** step may sound new. It is the process of finding **system specifications** by thoroughly analysing and understanding the **existing system**. We take a system that we know nothing about in terms of design, look at the **code** and try to find the **design** from the code. When we have the design, we can conclude the **specifications**.

## Reuse

**Software Reuse** can be done at several levels:

* **Application level**- Where an entire application is used as sub-system of new software.
* **Component level**- Where sub-system of an application is used.
* **Modules level**- Where functional modules are re-used.

### Reuse Process

There are two approaches to reuse, either we keep the **requirements** the same and **adjust components**, or we keep the **components** same and **adjust requirements**.

* **Requirement Specification** - The functional and non-functional requirements are specified, which a software product must comply to, with the help of existing system, user input or both.
* **Design** - Basic architecture of system as a whole and its sub-systems are created.
* **Specify Components**- By studying the software design, the designers segregate the entire system into smaller components or sub-systems. One complete software design turns into a collection of a huge set of components working together.
* **Search Suitable Components** - The software component repository is referred by designers to search for the matching component, on the basis of functionality and intended software requirements.
* **Incorporate Components** - All matched components are packed together to shape them as complete software.