Chapter seven
Software Maintenance

7.1 Software Maintenance

It stands for all the modifications and updates done after the delivery of software product.

Why modifications are required:

a) **Market Conditions**

تشمل السياسات Policies التي تتغير بمرور الوقت وال constraints الجديدة

b) **Client Requirements**

قد يطلب الزبون اضافة وظائف جديدة الى ال software

c) **Host Modifications** - If any of the hardware and/or platform (such as operating system) of the target host changes, software changes are needed to keep adaptability.

d) **Organization Changes**

اذا كان هنالك اي تغيير على مستوى العمل مثل تغييرات ادارية او انخراط الشركة في اعمال جديدة وبالتالي قد تظهرالحاجة الى التعديل على ال software

7.2 Types of maintenance

The maintenance made to the software may be:-

- **Simple changes** to correct coding errors

- **More extensive changes** to correct design errors, or significant enhancements to correct specification errors or accommodate new requirements.

Types of maintenance:-

a) **Corrective Maintenance -** This includes modifications and updations done in order to correct or fix problems, which are either discovered by user or concluded by user error reports.
Fault repairs Coding errors are usually relatively cheap to correct; design errors are more expensive as they may involve rewriting several program components. Requirements errors are the most expensive to repair because of the extensive system redesign which may be necessary.

ويشمل ذلك التعديلات والتحديثات التي تمت من أجل تصحيح أو إصلاح المشكلات ، التي يتم اكتشافها من قبل المستخدم أو متضمنة في تقارير أخطاء المستخدم.

يعتبر تصحيح Code errors عادة رخيص، بينما تصحيح design errors يعتبر غالي لانه يتضمن اعادة كتابة لعدة مكونات في البرنامج بينما Requirements errors تعتبر الاغلى لانها تتضمن اعادة تصميم النظام .

b) **Adaptive Maintenance (Environmental adaptation) -** This includes modifications and updates applied to keep the software product up-to date and tuned to the ever changing world of technology and business environment .

ويشمل ذلك التعديلات والتحديثات المطبقة للحفاظ على software product مواكبا للتحديثات التي تظهر في عالم التكنولوجيا وبيئة الأعمال المتغير باستمرار مثل تغير في (hardware or the platform operating system).

c) **Perfective Maintenance (Functionality addition) -** This includes modifications and updates done in order to keep the software usable over long period of time. It includes new features, new user requirements for refining the software and improve its reliability and performance.

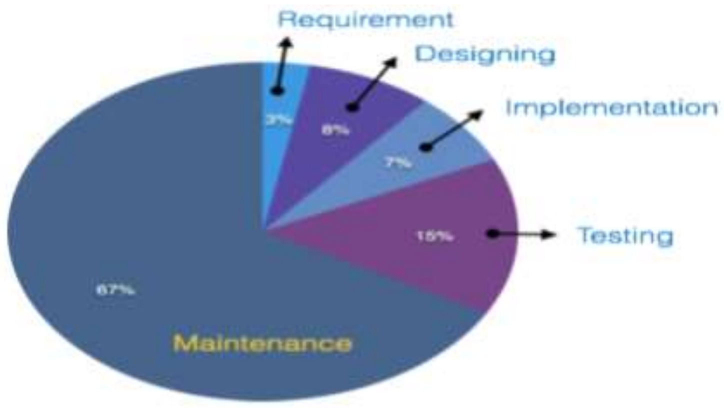
ويشمل ذلك التعديلات والتحديثات التي تمت من أجل الحفاظ على software قابلاً للاستخدام على مدار فترة زمنية طويلة .ويشمل اضافة ميزات جديدة ومتطلبات مستخدم جديدة لتحسين software وتحسين موثوقيته وأدائه .

d) **Preventive Maintenance -** This includes modifications and updates to prevent future problems of the software. It aims to attend problems, which are not significant at this moment but may cause serious issues in future

ويشمل ذلك التعديلات والتحديثات لمنع حدوث مشاكل لل software التي هي ليست مهمة حاليا ولكنها قد تسبب مشاكل خطيرة في المستقبل.

٧,٣ Cost of Maintenance

Studies agree that the cost of maintenance is high. A study on estimating software maintenance found that the cost of maintenance is as high as 67% of the cost of entire software process cycle.



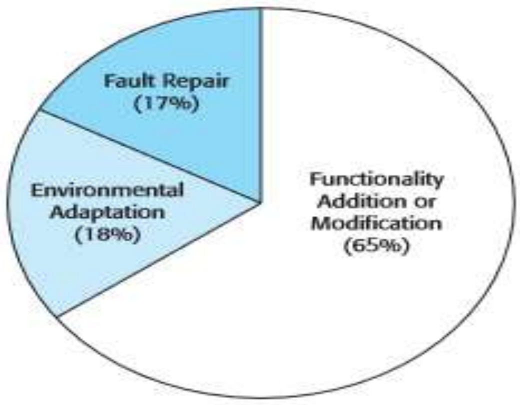


Figure (7-1) :- (a) Maintenance cost (b) approximate distribution of maintenance

There are various factors, which trigger maintenance cost go high, such as:

- **Program age and structure,** as changes are made to programs, their structure tends to degrade. Consequently, as programs age, they become harder to understand and change.
- As **technology advances**, it becomes costly to maintain old software.
- **Team stability**, most maintenance engineers are new and do not have any background about the system. They need to spend time for understanding the existing system before implementing changes to it.
- **Changes are often left undocumented** which may cause more conflicts in future.

7.4 Software Re-engineering

software re-engineering is updating the software to keep it to the current market, without impacting its functionality. It is a thorough process where the design of software is changed and programs are re-written.

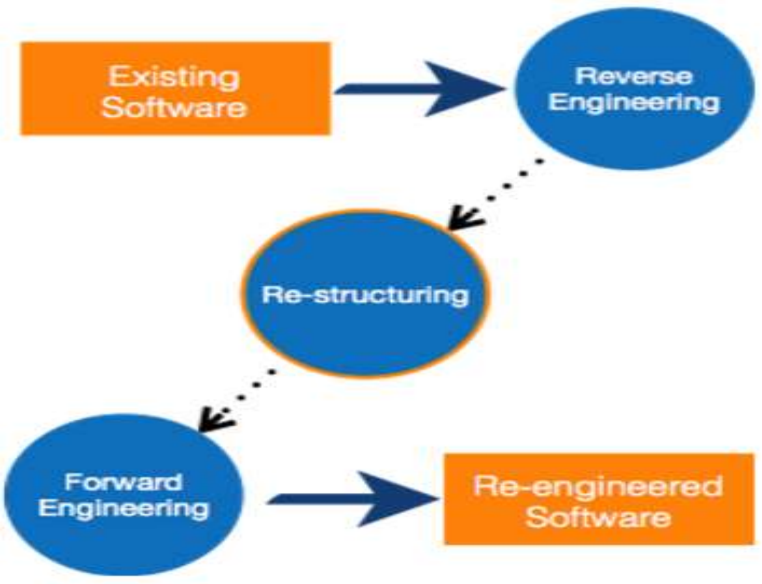


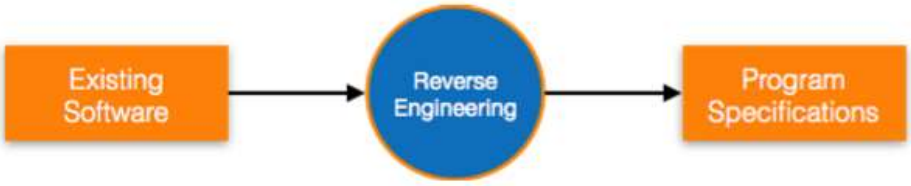
Figure (7-2):- Re-Engineering Process

- **Decide what to re-engineer**. Is it whole software or a part of it?

Then translate the source code where the program is converted from an old programming language to a more modern version of the same language or to a different language.

- **Perform Reverse Engineering**, in order to obtain specifications of existing software.

This process can be seen as reverse SDLC model, i.e. we try to get higher abstraction level by analyzing lower abstraction levels.



- **Restructure Program if required.** T

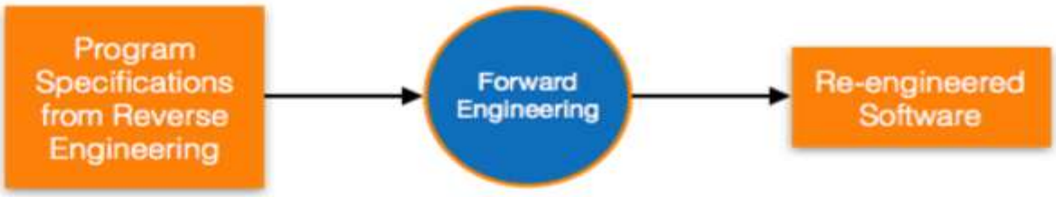
The control structure of the program is analyzed and modified to make it easier to read and understand, For example, changing function oriented programs into object-oriented programs.

- **Re-structure data as required.**

The data processed by the program is changed to reflect program changes. This may mean converting existing databases to the new structure. And finding and correcting mistakes, removing duplicate records.

- **Apply Forward engineering** concepts in order to get re-engineered software.

Forward engineering is same as software engineering process with only one difference – it is carried out always after reverse engineering.



7.5 Component reusability

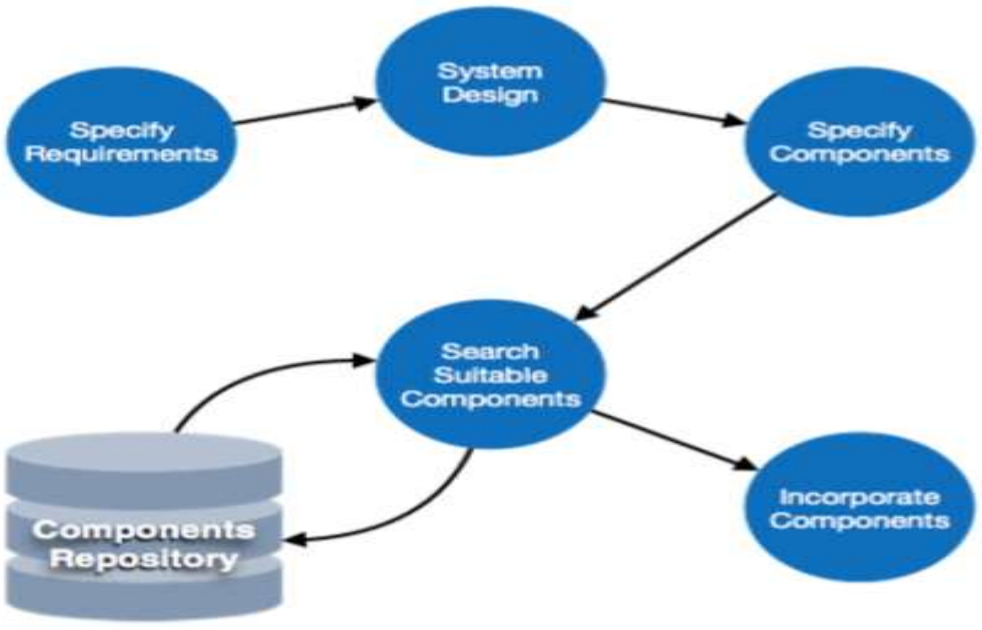
A component is a part of software program code, which executes an independent task in the system. It can be a small module or sub-system itself.

<u>Reuse Process</u>

Two kinds of method that can be adopted: either by keeping requirements same and adjusting components or by keeping components same and modifying 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 -** This is also a standard SDLC process step, where requirements are defined in terms of software language. 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.



- **Figure (7-3):-Reuse process**