Volume 8, No. 3, March – April 2017
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info
ISSN No. 0976-5697

Software Maintenance: Challenges and Issues and Models for Reducing the Maintenance Cost

Shray Khanna
UG, Depart of Computer Science
Vellore Institute of Technology
Vellore-632014, India

Ashay Shah
UG, Department of Computer Science
Vellore Institute of Technology
Vellore-632014, India

Shubham Jain
UG, Department of Computer Science
Vellore Institute of Technology
Vellore-632014, India

Ramanathan L
Assistant Professor, Department of Software Systems
Vellore Institute of Technology
Vellore-632014, India

Abstract: Software maintenance is a very tedious and vital job in software development life cycle (SDLC). In today's ever growing technological market outsourcing is done for best product delivery as well high efficiency. The outsourcing companies usually deliver a completed model to the company or user and further evolution or changes are done according to the requests made, these evolution or changes comes under Software Maintenance. Software maintenance is basically the modification done to assure a quality product after it is sent to the company or user who ordered it. This paper discovers the current models and strategies taken up by most of the leading companies for software maintenance and handling. It also explores the common challenges faced and its mitigation strategies. The preventive strategies are then discussed to help reduce for the issues faced by companies so as to overcome the overhead cost after delivering the product. The model/framework and strategies will explain the working into different stages and how the traditional methods can be used effectively. The maintenance issues is a wide field where each product has its own issues while there are some general issues too. This paper focuses majorly on general issues of software maintenance.

Keywords: software maintenance, software development life cycle, challenges, model, mitigation.

I. Introduction

Building up of software goes through many processes. Each step is very crucial and exclusion of a single step can lead to a faulty software or errors in further steps. The software goes through a cycle called SDLC or software development cycle in which each stage is defined for its usage in the completion of the product. A company usually outsources the components of each stage to get a detailed data for building up of the software. The outsourced companies usually specialize in a particular field such as surveys, reports or ground testing [4]. These components are then combined as per the architecture of the software given by the company and specified by the user. The completion of these components is very important for the initial setting up and testing. A SDLC includes communication, requirement gathering, feasibility study, system analysis, software design, coding, testing, integration, implementation, operations, maintenance and disposition [2].

Communication is the initial stage where the product is ordered with vague specification depicting how the software product should be and usually the terms are negotiated on various factors. Requirement gathering is the second step where a team is formed from various departments to get detailed information in the problem domain. The requirements are first listed through various brain storming sessions and segregated into three types of requirements that is user, system and functional requirements. The requirements are gathered according to the type of product by performing different task as per user. The feasibility study is a step when the team comes up with a rough idea of how the software product is to be developed and whether or not all the requirements can be fulfilled. System design helps to ensure the best fit of software models into the rough design. It includes system limitations and problems which might be faced while developing.

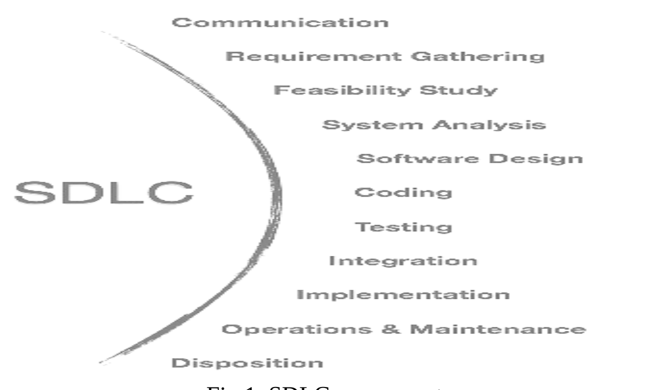

Fig 1. SDLC components

The next step is software design in which all the knowledge gathered from analysis and requirements phase are put into use by building the basic design of how the software should look and respond. The coding phase is basically the implementation of this software design on different platforms and frameworks using different languages required as per the design. Testing is very important to know the faults and working of the software created. There are several tests to ensure perfect working of the software and the software is tested usually by coders itself. After coding the product gets integrated with different libraries, databases and other programs for integrating it with outer world entities as per the design configuration. The implementation stage ensures installing of the newly created software into user machines for final output and testing. It is then tested for non-functional requirements. In operations and maintenance phase the product is upgraded or evolved after the delivery to increase its efficiency and keeping the program up to date. The final stage is disposition where software declines due to performance issues after ongoing usage over time. It results in closing down the system or archiving data [1].

Software maintenance is very crucial for finishing and upgrading the product to increase its efficiency and make it less prone to errors. Software maintenance also helps to bring the product as close to user specification as possible. Software maintenance modifies the software after delivery of product. The modifications are usually required because of the ever changing market trends, client specifications, the host changes in hardware or platforms used if necessary and organizational changes. The maintenance is divided up into different types as per the nature of product. The types of maintenance are corrective maintenance, adaptive maintenance, perfective maintenance and preventive maintenance. The corrective maintenance gives modifications and updates done to fix the problems or correction of products as processed through error reports or by user. Adaptive maintenance is updating and modifying so as to keep the software product up-to date in order to keep up with present trends and scenarios. Perfective maintenance is modifications and updates such that the product runs over a long period. This usually includes new features and requirements to enhance the working and dependability of the product. Preventive maintenance includes updating and modifying so as to prevent future failures or errors in the product. It takes in non-functional and functional requirements which are not significant in present but may cause errors or problems in future.

Software maintenance ensures continuity in service so as to keep the product in market as long as possible. With change in trends and development of new technologies it is important to modify and update the software according to the present needs. Proper maintenance ensures all these by taking into consideration each factor for maintaining the continuity of the product.

II. MAINTENANCE ISSUES

The cost of maintaining and processing a software product is pretty high. Recent studies and reports show that the cost of maintenance is 50% more than the entire cost of developing the software [3]. The various reasons which makes the cost go high are usually divided into two categories which is real world problems and software end issues.

The real-world problems are always kept in mind while developing but the actual affects take place in the maintenance stage [3]. The major issue is that the software usability is up to a particular extent after which the product cannot be upgraded further and has to be closed. The old software requiring less configuration to run are not able to perform properly on the new environment provided to them. The trial and errors methods used while developing a software is an issue and should not be encouraged. The new changes can often disrupt the original design and structure which makes it difficult for subsequent changes to be places. The changes can often be left undocumented which causes problems in future while maintaining. The software end issues can come up at any time due to various reasons in platforms or user machines [4]. The structure of the program might not fit into the newly updated system. The programming language which is not common may cause an issue while performing efficiently. The software depends on the external environment such as hardware and other devices plugged in, if any one of them causes problems or is faulty then the software might not respond well or might not even work [1]. The reliability on the original structure while keeping it up to date is very difficult. According to these types the following issues might occur:

A. Program Workability

Program is developed by keeping future issues into consideration. The modifications and updates can only be made by maintenance engineers if the original program is re-usable. The most problems that occur are generally in this phase and is usually considered as central research area under maintenance. For a maintenance engineer it is important to understand the behavior of code, functionalities and how adding new features might change the working. Once that is achieved it is easier to do the necessary changes. This field consumes most of the resources for handling and maintenance jobs [2].

B. Impact Analysis

The maintenance analysis is a tedious job as we have to consider the previous analysis done and incorporate features according to the new one. The maintenance analysts face the general challenge that is to determine how the changing of system affect the working environment of a software and what are the impacts on the jobs. The main job for the analyst is to analyze perfectly for the part that is to be modified. The impact analysis is thus the analysis done to minimize the risks of errors in working while adding new features to a software product [1].

C. Implementation Modifications

Changes in the implementation take place over different stages. The architectural design is divided into different components and each component has its own implementation technique. One problem faced is that if a component is upgraded or modified it may not respond properly or may not fit in the original design [6]. If that happens design has to be changed entirely or component has to be brought back to its original state or the nearby components should be changed as per the specifications and the architecture. This process of changes is referred as Change Propagation. Consistent software usually includes successful changes while change propagation occurs when the software has inconsistencies.

D. Regression Testing

Regression testing is applied after updating and modifying the software product to know how changes work in real environment. This testing assures that the modifications done are not faulty and the system architecture has not been affected by changes done. The testing can be done several times to assure quality product and better performance. The system software changes if there is any addition of module to the system [4]. New rules have to be defined, new control system and new data flow paths are sets with changes in input /output. The new modification might cause an error with the previous system and devices. Regression testing is thus used to make sure that the changes should not affect the original system architecture and there is no hindrance with the working. By doing testing successful tests are acknowledged and the modifications are passed accordingly. Regression can be done manually by the re-execution of some previous test cases and using the previous test sets to ensure proper working. The new test cases are added later to make sure the modifications are valid and efficient. It can also be done through an automated way that is by using capture/playback tools. The tools are used for testing and capturing of test cases by the maintenance and software engineers [5]. There are three different classes of test cases that is the software functions which are needed to be exercised on the test cases must have a sample, new additional software in the system affect the change and implementation and finally the components that are considered for tests change according to the modifications and updates. The tests usually become large as the test sample size increases.

E. Programmer Availability

The availability of programmer is also a major concern. The original version of the code written cannot be easily edited by a new a programmer as the knowledge and understanding of previous used components is very essential. The response requests of programmers should be managed according to the updates and modifications required [5]. The previous version of code might not be easily understandable and re-usable and might not fit for the new modification. The programmers should keep all these things in mind while modifying or developing a new code for a component or system.

F. User Knowledge

Many problems are faced due to the scenario where user lacks the knowledge required for understanding the system. This limits the capability and working efficiency of the system and brings down the product value. This can be also because of lack in training provided or taken up by the user. It is important for a user to understand the changes and updates done on the system. Major effort is put in by the maintenance team so that the users are able to understand the system and limit the risk of bringing down the changes. The limited understanding concludes low quality documentation and descriptions of the updates. To minimize all the limitations different strategies are taken up like boot camps or different learning ways to ensure that the user understands the changes and responds well to the system changes.

G. Demands And Expectations

These refer to both the market and user demands and expectations. The demands by user are generally to change the software as soon as possible to increase its performance as per the new trends. The market or in real world the timely changes and modifications are required periodically to keep up with new advancements. The company is expected to ensure the proper working of software product according to the new demands and expectations and are required to keep up with the new market trends. Due to constant changes in technology sector, the burden on maintenance team increases as the demands of both increases [6].

H. Software Product and Real Time Environment

The modifications sometime work perfectly at the time of testing but may cause trouble at real time where the data flow cannot be controlled and the data to be worked on forms wide cases [4]. This happens when there is partial testing or the test case sets are taken by single resource. This form biasing at the time of test and the results may come only for a particular type of scenario. The relationship between the software product and the environment should be assured so that the system works on any form of data [6].

I. Maintenance Team and User

The major cause of problems faced by the companies are due to lack of interaction between the user and maintenance team which leads to poor quality surveys and make the product more vulnerable. The user knowledge also is a factor between the relationship of maintenance team and the user. The user specifications should be handled according to the modifications which can done to the product and based on the survey the reports should be sent [5].

J. Databases

Databases play a vital role in functioning of any software product. The size, value, attributes, schemas and relationship all contribute towards the efficiency of system. Database size is an important factor which has the number files and characters stored. Database size increases or decreases the speed of functioning of a software. It may also happen that modifications done on a product may lead to altering of the database. Databases are thus carefully administered by an administrator who keeps a check on the database such that any alterations or update might not corrupt the data or to check the data losses due to modifications [1].

K. Product Quality

It is very important to ensure that the delivered product is of top quality. Any maintenance done on that product should be ensured with proper care so as to maintain quality. Quality of a product includes all the stages that is ranging from data and sources collected to combining of different components to form a product. The code on which the program runs may have some issues or ay run on a particular type of data sets. The non-functional requirements may not be fulfilled after the modification of the product or the new code changes may affect the working of the product on some data points. The quality is then compromised due to the above factors.

L. System Age

Each software product has an expiry date associated with it that is the time when that system design may become obsolete. A time comes for each product when even the modifications and updates doesn’t increase the performance and efficiency, at this time the software system is shut down by the user or company because investing further is of no use. The age factor is correlated with corrective maintenance of the product.

M. Staff Dependencies

The team has a major contribution in maintenance as well as building up of the product. The general problems faced by any company is that the turnover from the staff makes the maintenance improper. Improper maintenance means difficult handling of the product as the original team who knew the product from top to bottom are not present so the new team has to learn and understand each step from its making to the recent modifications which causes faulty new updates. The effort spent on understanding is far greater than that of modifying. Staff turnovers is the major problems faced by the manufacturing companies and thus maintenance cost can increase due to this [5].

N. Operating Environment

The operating environment includes all the components of the ready product and the testing of the product depending on different specifications and test cases. It includes the hardware and software reliability, failure in tests, documentation and integration of the product with real environment [4]. The operating environment is made in a way such that the real-world application is as close as possible so as to mitigate errors. The problems generally faced are that of the working of the system according to the new trends and failure of the teams to replicate the real-world issues into test cases. If the testing part results failure then the operation has to be performed from the start. Another issue faced is in documentation where each new modification and update has to be carefully listed so that the user is able to comprehend with the new changes. If there is any issue in the documentation then the client connection may become vulnerable thus the documentation is very crucial at the final stage [6]. The documentation hence should be of high quality but sometimes negligence from the team makes the document more difficult for the client to understand. A large system with many components usually increases the chance of errors and the issue comes to the databases which are relatively large as there is a great change in the system files and properties. These operations are carefully processed and thus they all come under the operating environment.

O. Maintenance Budget

The budget given by the company for a particular product sometimes seems less and even though the team puts all its efforts in maintaining, the job is not properly accomplished. The resources associated with the product directly affects its maintenance, if there are plenty of resources given by the company for a particular product then the maintenance jobs are performed perfectly with minimal errors.

The maintenance issues are sometimes complex to handle and even though the resources provided are in plenty the product may not perform as per the new specifications. To minimize the errors every company follows a particular model which is best suited for the product. The aim is to make sure that the model fits the product and other efforts are put in to lower errors such as re-documentation, testing qualities and various other features which are explained in the following sections.

III. MAINTENANCE ACTIVITIES

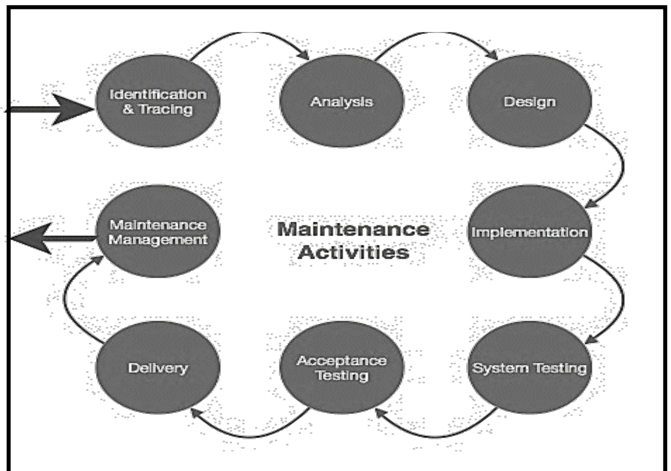

Fig 2. Maintenance Activity model for simplification of jobs

Almost all companies follow this framework as the base for keeping the check of maintenance activities. This sequential framework is provided by IEEE to ease the flow of activities and at the same time taking into consideration the jobs done under each. The process demarcates one maintenance job into several steps which are in sequential order such that start of the job is correct and minimum roll back is required. This is a basic framework which can be used iteratively for different activities and can be easily expanded as per the specifications and processes [2].

A. Identification and Tracing

Identification of the problem and tracing its root is the initial step and a very crucial one. The team looks into each perspective very carefully so that the problem identification is done with no lags. The modifications or maintenance requirement is identified by a set of steps according to the specifications of client or the periodic change of product. The major issue for maintenance is generated by the user via report logs or messages. The type of maintenance on which the team should work is also identified.

B. Analysis

The modifications or updates in the system should be analyzed with proper care and handling. The analysis is done so as to make sure that the new changes are not impacting the system and it is not making the system vulnerable to attacks. The analysis ensures the safety and security of the system. If the impact of the change is drastic then more number of options are looked into so as to get best solution from the problem. Once solution is found then the required changes to be done is materialized into the specifications received. At this stage only the cost of maintenance is analyzed and an estimation for the job is provided.

C. Design

The designs which are required for the new components are created at this stage and it is made sure that it fits in the original system. If the new design causes problems in the working or changes the desired working then that design is discarded and the team has to work on the new design. The team makes sure that the new designs fulfills all the requirements and specifications given by the user. The test cases for validating the new designs are taken to ensure high performance and efficiency of the system under the new design.

D. Implementation

The new designs are forwarded to the programmers who code for the implementation of the new design making sure that it follows the original structure of the code. The testing of each component is done in parallel to ensure the working of new component with the original code. The programmers ensure the optimality in code so that the system performs with more accuracy in the new environment.

E. System Testing

Integration testing is performed among the newly developed components. The testing is also carried out at both system and component level. The integration tests are also carried between the system and each component to ensure that each new module works well with the system with no errors. After integration testing regressive testing procedures are carried out on the entire system to note its working with new components. The tests are carried out in the operating environment to ensure that it works in real time with minimal faults and at high rate [2].

F. Acceptance Testing

After testing it in the operating environment the system is tested for acceptance with users to ensure that all requirements are fulfilled as specified by the user. If there are any discrepancies at this stage then the user's evaluation is noted and rectified by following iteration. The iteration performed is then made with user so that the acceptance from the user is achieved.

G. Delivery

After getting acceptance from the user for the new changes the new product is deployed to the field by different methods. Generally, the new updates are given through an update package or re-installation or up-gradation of the entire software product. The final tests are carried out by the client now so as to ensure that all non-functional and functional requirements specified are fulfilled [7]. The entire system is documented with incorporation of new changes/modifications made on the system. The report and documentation is given to the client.

H. Maintenance Management

The team assigned now solves the queries of different users who face problems with the software either in installation or working. The queries are solved for the user who face difficulties at any step and a proper log is maintained so that any errors happening are rectified at the next modification or update to minimize bugs.

IV. MAINTENANCE REDUCTION

There are various ways involving the reduction of the software maintenance. If there earlier steps are carefully looked out and if they are done with high precision then maintenance jobs can be significantly reduced. Jobs like designing the right architecture, programming steps, appropriate and unbiased testing and proper documentation of the product. The following methods describe the possible ways in reducing the maintenance:

A. Re-documentation

The maintenance jobs are can be reduced almost up to half if the Re-documentation of the product is done properly in accordance with the new trends and specifications. It is noted that the re-documentation reduces the cost by almost 12%, which is a very big number as companies invest a huge sum of money in maintenance of the products. It is then made sure that the software teams should be able to know and make proper use of the legacy of the system. One purpose is to generate the documentations for all the new tasks carried out and to improve the current provided document since updating system changes is a tedious job. It is also done to create alternative opinions of the system in dataflow, design or control flows to make sure that the users are able to understand the product easily. The re-documentation can be incremental re-documentation or model oriented re-documentation [2].

Generally incremental re-documentation is preferred over model oriented. The incremental re-documentation improves the understandability of product and incrementally rebuilds the document after the jobs of programmers are completed. The major advantage of this is to take out the common issues in the maintenance process and makes sure that the requirements or specifications made by the user or client is fulfilled. The initial step is to request for the changes that need to be collected by the customers [1]. The change requests made should understandable by the programmer so that the necessary changes take place successfully, for this the current documentation should be perfectly made and should be easy to understand. Programmer then implements the specified changes according to his knowledge and tests the compatibility of the new components with the architecture. The programmer also measures the correctness of the program and confirms the changes made. At the end of re-documentation PAS tool is used for program comprehension achieved during the implementation of the new components and it is recorded in hypertext so that the changes are not forgotten and can be used in later implementations.

Model oriented documentation uses a different strategy in dealing with this as it uses source code to re-document the legacy of system. Model Oriented Redocumentation (MOREDOC) generate models from the legacy systems that were previously produced and generates new models from the previous structure by using Model Driven Engineering (MDE) technique [1]. Software models are able to bridge a gap from an old system to an updated one and hence enhances the abstraction level in documentation done. This increases the computerization in the development of a program. MOREDOC hence works to get all the information about the system and ethics put in using MDE to generate a document which is well defined and includes everything from the legacy system source code to new component implementation.

B. Decreasing Turnovers

One way to reduce the cost of maintenance cost is by reducing turnovers internally and externally. The internal turnovers can be due to many factors happening inside the company and may result in a huge turnover while external turnovers are due to the external factors and usually influence the employee by additional benefits or services being provided to them. To reduce this completely strict policies and work schedules are made such that in a project the turnovers are minimal [7].

C. Dead Code Elimination

Dead code elimination makes the system more optimized for running and decreases the failure rate. Dead code is basically a chunk of code which is not used by the system and has no role to play in any component [7]. This code is generally there because it was used in the initial steps of the product constructions and after modifications and updates the programmers didn't remove it from the system. Reducing code size impacts the speed and performance of the working as the system uses less resources and thus runs at a higher speed [1].

D. Understandability

It is important for the maintenance team to gain a proper knowledge and understanding of the system before implementation so as to minimize the risk of errors. They need to go through all the phases of making of the product and should be updated about the current market demands and how modifications need to be implemented. They need to ensure that they understand the design of the system very thoroughly and the interconnections of the components for various purposes is done with par excellence. Changes that need to be done on the design need to be well communicated with all the different teams and individuals such as programmers and users. The documentation should be made instantly such that information about each new component added or modified is maintained. The documentation made should be accurate and well organized with all the specifications and detailing. The programmer needs to ensure that the tests that are been carried out are performed under all the possible scenarios and its real-time implementation should be projected. It is necessary to make sure that there is no redundancy in the data [7].

E. Software Re-engineering

The updates and modifications done on a product without affecting its functionality is called software re-engineering. It is a major process carried out by all the firms to eliminate the bugs and errors reported by the user. The code is re-engineered such that it works well with all the platforms. Legacy software cannot be tuned every time as it may become obsolete due to the new changes in the market and thus updating might seem a more tedious job than developing a new one. One thing that doesn’t change is its functionality although it may become old and not up to date but the base functionality of a product never changes.

The maintenance team also focuses on the components that need the most modification jobs to make sure the product is up to date with the market and thus re-engineering is done to ensure the workability of these components.

Re-engineering process include deciding the part to be re-engineered. Performing reverse engineering on the system so as to obtain existing specifications of software. Restructuring of the program wherever necessary and restructuring of data according to new specification and market demands. And final step is to apply forward engineering concepts so as to get the re-engineered software [2].

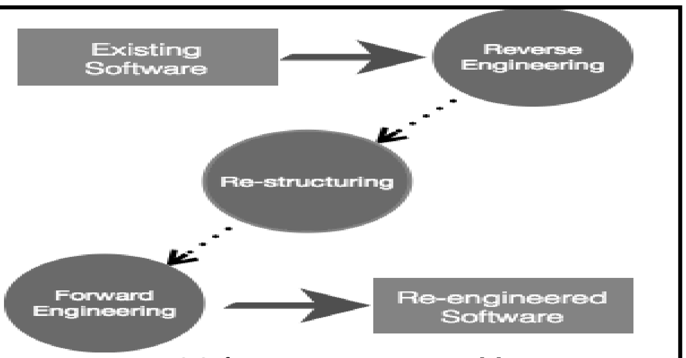

Fig. 3 Software Re-engineering model

F. Reverse Engineering

Reverse engineering is a process to complete system specifications by proper analysis and understanding of the system. This process is sometimes considered as reverse SLDC model since the abstraction is done on a higher level by analysis of the lower ones. An existing system is then first implemented about which the team knows nothing about, this is done to get to know the functionality so that it can be implemented together with the current product [3]. Designers of the team then carry out reverse engineering by seeing the code and converting it into system design and the specifications are usually concluded at that time only.

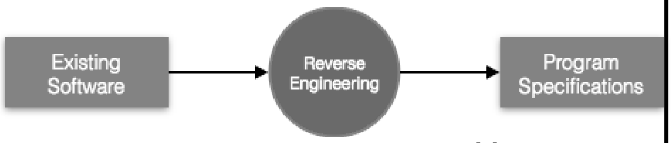

Fig. 4 Reverse Engineering model

G. Program Restructuring

The process of restructuring and rebuilding of the existing software is known as program restricting. The programmers re-arrange the code into the same language or a different one as per the information provided to them. Restructuring includes program and data restructuring but it is not mandatory to include bot in the process as it is according to the specifications and rules provided. Restructuring increases the non-functional requirement of the system and the functionality is not affected by the changes. The bugs and errors reported in the product are rectified at this stage [3]. The dependability of a software on obsolete platforms are also removed by carrying out restructuring process.

H. Forward Engineering

It is the process of obtaining the desired product from the in-hand specifications and data which was obtained during the reverse engineering process. It is assumed that some processes of software engineering have already been carried out. The working of forward engineering is very similar to software engineering process with one minor difference that is it comes after the reverse engineering process [3].

Forward engineering not only covers the previous design and functionality but it makes sure that all the current and future prospect designs are considered so as to enhance the quality and functioning of the system.

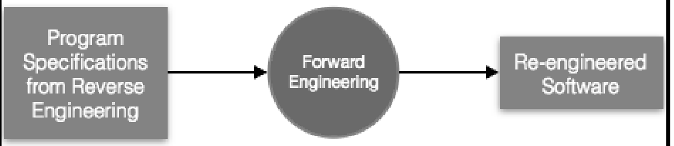

Fig. 5 Forward Engineering

I. Reuse Process

The reuse processes are carried out by most of the maintenance team as it is easier to reuse a product or a component rather than developing one. Generally, the methods adopted are by keeping the same requirements as before and adjusting of components or modifying the requirements and keeping components same as before [2].

The processes carried out in the reuse process includes requirement specification of the product according to both functional and non-functional requirements in accordance with the existing system. The design is a major step where the errors can come up and if not done with precision might cause harm to the system. The design process is carried out similar to SDLC and basic system with the new component as a whole is created with its sub-systems. Specifying of Components by studying the design created and dividing the system design into various small sub-systems or components on which various sectors of team can work on. Searching of the suitable components from the repository is essential to get the correct match for components on the basis of specifications and functionalities provided to them.

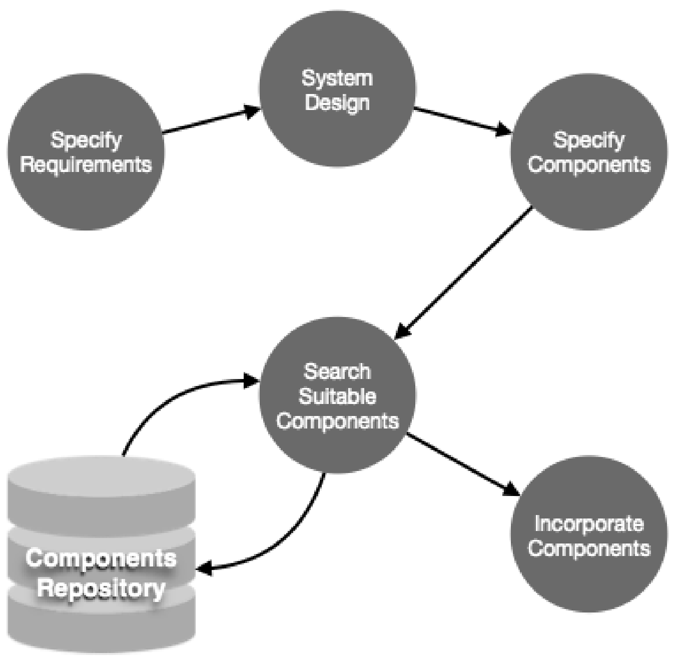

Fig. 6 Reuse Process model

Components are finally incorporated as a single system to be able to work in real time and is tested according to the test sets. The product is now updated and can be made available to users.

V. CONCLUSION

The software maintenance is considered as the major process after the product is made and carrying out jobs can become tedious. Software maintenance is an ever-growing market where the money and cost for maintenance and production is usually more than that of developing. The processes carried out are usually taken up by a team and divided accordingly as per the specialization of team members and specification requirement. The processes done in modifying and updating a software needs to be carried out with proper precision such that all the previous bugs are rectified and the old functionality is not compromised. This is made sure by carrying out different processes by taking up a model so that there is no confusion in the maintenance jobs. The maintenance part of the product is very essential to keep up with market such that the system doesn't become obsolete. The challenges or issues which occur after the delivery can be due to the problems in the development phase or can be due to changing technology. The problem is taken up and tasks are performed such that the costing doesn't go very high and the base functionality is maintained and shouldn't be obsolete. The processes taken up by the models of maintenance are thus handled by the team and updates of the system are made available to the user with bug or error free product.

VI. REFERENCES

[1]Uttamjit Kaur, Gagandeep Singh "A Review on Software Maintenance Issues and How to Reduce Maintenance Efforts" GIMET, Amritsar. Volume 118- No. 1, May 2015.
[2] (Online Source) https://www.tutorialspoint.com/
[3] Tracy Hall, Austen Rainer, Nathan Baddoo, Sarah Beecham "An Empirical Study of Maintenance Issues within Process Improvement Programmes in the Software Industry" Department of Computer Science, University of Hertfordshire, UK.
[4] RanaEjaz Ahmed "Maintenance issues in outsourced software components" School of Engineering, American University of Sharjah, United Arab Emirates.[5]Aakriti Gupta, Shreta Sharma "Software Maintenance: Challenges and Issues" St. Xaviers College, Jaipur, India. Vol. 4 No. 01 Jan 2015.
[6] Rajiv D. Baskar, Srikant M. Datar and Chris F. Kemerer "Factors Affecting Software Maintenance Productivity: AnExploratory Study" Carnegie Mellon University and Massachusetts Institute of Technology.
[7] Chris F. Kemerer, "Software complexity and software maintenance: A survey of empirical research" Sloan School of Management, Massachusetts Institute of Technology, Cambridge, USA.