Concerns in Maintaining Reusable Software Components and the Possible Solutions

Anshul Kalia¹* and Sumesh Sood²

¹Department of RIC, IKG Punjab Technical University, Kapurthala – 144603, Punjab, India; ansh.kamal007@gmail.com
²Department of Computer Applications, I. K. Gujral Punjab Technical University Campus Dinanagar, Dinanagar – 143531, Punjab, India; sumesh64@gmail.com

Abstract

Objectives: The subject of the study deals with the problems faced during the maintenance of reusable software components. It aims at exploring different problems, and their potential solutions. Methods/Analysis: The study scrutinizes the previous methods that were followed for the maintenance of reusable software components. It takes note of the standards that are framed for maintenance purposes. It then figure out the major problems that exist in maintenance of reusable software components. It explores the present development scenario of reusable components to reach out to the solution and determines how tools can be helpful to address the problem. Findings: The research enforces to evolve the relationship between maintenance and reusable software components. It comes up with the problems such as change control, version control, scarcity of traceability, outdated legacy systems, lack of adequate documentation of reusable components. Consequently, the solution to the concerned problems have evolved in a form that is represented as notification about changes, integrated environment of reuse repository with SCM, reverse engineering as tracing tool, upgrading through re-engineering, changes should be reflected in documents. The study also presents a set of tools that may help in conquering the problem. The analysis of previous work is significant in establishing the consensus about the stated problems and mentioned solutions. The findings have their significance in the fact that they have been addressed in an integrative manner. The set of tool support adds value to the study. Application/Improvement: The scope of study can be enhanced by taking a case to endorse the findings.

Keywords: CASE Tools, Maintenance, Reusable Software Components, Solutions Concerned, Threats To Maintenance

1. Introduction

As per IEEE standards definition the maintenance of software is defined as "Modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment" and the set of activities that takes place such that software installed for operational use continues to perform as intended and fulfill its intended role in system operation. Software maintenance includes improvements, aid to users, and related activities (The IEEE Standards Dictionary: Glossary of Terms Definitions, IEEE Std 610.12-1990). The same definition will apply to software components as well. Reuse is one of the traits of a software component. Thus, it can be stated that the maintenance of reusable software components is concerned with the activities which keeps the reusable components in the operational state and makes it to fulfill the intended functionality. Also it is concerned about the improvements, modifications, corrections, and assistance to users. The purpose of maintenance is to correct errors, to update the software system as per the future problem requirements, to enhance the functionality of the components, and to cope up with the changed environment. The software maintenance can be classified into corrective, adaptive, preventive, and perfective maintenance. The corrective
Concerns in Maintaining Reusable Software Components and the Possible Solutions

Maintenance refers to the correction of errors that arises after the software product is in use. The errors may occur anywhere such as in specifications, designs, coding, testing, and documentation. Adaptive maintenance refers to the modifications in the software system when there is a change in the external environment of the system. Preventive maintenance refers to the anticipation of future problem that may occur in the software and improving the system in such a way that these problems may not occur. Perfective maintenance adds functionality to the system by incorporating enhancements as requested by the client. Perfective maintenance is intended to improve the effectiveness and efficiency of the system. Most of the time and money of maintenance period is covered by the perfective maintenance.

Maintenance covers more than a half life time of software development life cycle for a software product and also the cost of maintaining software is more than that of its development. The traditional software was developed from a single vendor and the maintenance of such a product by the original developer is easier. But in case of component based development, there is a possibility that the software components do not belong to a single vendor rather it belongs to multiple vendors. In such a scenario the maintenance of components is a challenge for the professionals. From the past experiences, as in the case of Y2K problem, it is felt that the lifetime of a software component may exceed to its originally estimated lifetime. That is, if a component is performing to a satisfactory level and has a good track record of operational performance then there is no point of abandoning such a component and to incorporate the new one with unproven track record. So the maintenance of such components requires adequate technologies and methodologies to cop up with the issues.

2. Previous Work

A study about the maintenance was conducted by Kwon in 1997 in concern with the software reuse. The study stated a relation between software reuse and maintenance. It advocated for the integrated Software Configuration Management (SCM) environment for maintaining reusable components. It then implements the model named TERRA for managing and controlling the reusable components in SCM environment.

Voas in 1998 tries to explore the technologies that may be required to maintain the components based systems. It predicted that the maintenance technologies will be advanced in the future to an extent that the maintenance of components will be easier due the exchange between the components.

In 2004 discussed the issue of component based software maintenance. The objective of study was to resolve the maintenance issues through software configuration management techniques. The SCM techniques provide support to maintenance activities. It also discusses about the configuration management repositories and the component based artifacts.

Al-badreen considered the lifecycle of software components in 2011 from two sides, i.e. build for reuse and build by reuse. While considering the lifecycle it pointed out the need for formal methods and technologies to be adopted. It discusses about the development of a systematic framework that addresses the lifecycle of software reusability.

In 2012 Imeri puts a light between the connection of software reusability and cloud computing that can be established in many ways. The study talks of installation and usage of applications through cloud. But if considered in the context of maintenance it can be beneficial.

3. Relationship Between Reusable Software Components and Maintenance

The relation of reusable software components and maintenance evolves through reverse engineering and re-engineering. In reverse engineering, the programs are reverse engineered in order to understand the specifications and design secrets. It is a process of recovering design, architectural, specifications, and data related information from an existing system. Reverse engineering can be better understood with an example of a hardware unit. The hardware unit of a company can be reverse engineered by the competitors, which is, disassembled in order to determine how it functions, and to obtain the secrets behind its functioning. Same is the case of software but the reverse engineering in most of the cases is done by the developing organization of its own software units. In re-engineering, it recovers the design information of the existing system as well as it reconstitute the system in order to incorporate enhancements to increase functionality and to improve performance, and quality.
The maintenance and the reusable components are related to each other through reverse engineering and re-engineering. As with these processes the maintainers will be able to extract the existing components and will be able to reconstitute the existing components with enhanced functionality. It can be then stated that the processes of reverse engineering and re-engineering are utilized to perform the form of perfective maintenance. The reason is the perfective maintenance is intended to enhance functionality and to improve effectiveness and efficiency of the system\(^\text{[11]}\). The similar purpose is fulfilled by the process of re-engineering. The reuse processes 'development for reuse' and 'development with reuse' supports the maintenance process\(^\text{[12]}\). Also the processes of both, that is, reusable components and maintenance, require the software configuration management activities to control changes\(^\text{[13]}\). There are several points to establish relation between these two.

**4. Maintenance Issues Faced by Reusable Software Components**

Most of the problems related to maintenance have its origin in planning and development phase of software lifecycle. Indiscipline and lack of control in software development activities leads to maintenance problems in the future\(^\text{[11]}\). The maintenance problems faced by the reusable software components is identified and elaborated in this section. A large span of maintenance problems of reusable software components may exist. The study highlights the problems of common perspective. These problems are:

**4.1 Change Control of Reusable Components**

The problem of change control of reusable components is considered under the SCM discipline\(^\text{[6]}\). There may take place a lot of changes to a reusable component. The issue is to manage these changes in such a way that it should not affect the performance, functionality, quality, and effectiveness of the component. It states that how to control the changes in a component that the component itself and the projects with which it has been integrated are least affected by it. Some of the components are designed in such a way that they are unable to accept modifications unless the modifications are made in their specifications and designs. Modifications in such type of components are error-prone\(^\text{[14]}\).

There is also a possibility that a single component may be used in multiple projects. Every project has different context and has its own functional environment. A single change or a change in a single line of code may have different effects in different projects. The effect of change in that component should affect the projects adversely. It is very difficult to analyze the effect of change\(^\text{[6]}\). Communicating changes in reusable components to its different stakeholders is also an issue under change control. If changes are not communicated adequately then it too has a potential to affect the components and the systems with which they are integrated either immediately or later on\(^\text{[14]}\).

**4.2 Version Control of Reusable Components**

A version is defined as a particular form of something, differing in certain respects from an earlier form of the same type of thing\(^\text{[15]}\). Versioning is defined as a process of assigning unique names or numbers to unique states of computer software or software components and versioning is a kind of maintaining a product line of reusable components. The version control is referred to the management of changes in documents, computer programs, large websites and other collection of information\(^\text{[16]}\). Issue of versioning and version control will come under the configuration management.

A reusable component may have number of versions which may have variations to a certain extent and in certain respects from one another. This may lead to issues amongst the maintainers and reusers. If a reuser is using a version of a reusable component which is an older one, and now it wants to update the version of a reusable component to a newer one. Then there may arise an issue that the configuration of reuser may support only the older versions and the configuration of reuser is unable to support the newer ones. There is also a possibility that the maintainer is unaware of the newer version of a reusable component or the information of a new version is not communicated to the maintainers. Also the issue with maintainers is that while the maintenance process they will have to identify which version of a reusable component they are maintaining. Also tracing the origin of a number of versions can be difficult throughout many versions if they are not properly documented\(^\text{[6]}\). If different versions of a component in different projects are being maintained by a single maintainer then the maintainer will have to keep track of changes in different versions which are a difficult task. Ignorance or
leniency while maintaining versions can lead to serious problems in a project.

4.3 Scarcity of Traceability

Traceability of code of reusable components is another problem that is faced while maintaining the reusable components. There is very less possibility that the code of a component can be traced to its specification details and design details. A maintainer can look into a source code to track the errors and defects. But in the absence of source code it can be difficult for the maintainer to track the errors and defects in the components. Sometimes also there is a code that is not accompanied by the code comments or the code comments are not adequate to specify the purpose.

To understand the code that is developed by someone else can be difficult and if that code is understood in a wrong direction, severe problems may occur. To track errors in a program developed by someone else requires understanding of that program. There may no one around who can explain the program or has been there when the program was being developed or part of developing team.

4.4 Outdated Legacy Systems

The problem of maintaining outdated legacy systems or components is there. Some of the software components were not developed with a mindset that they can exceed their estimated lifetime. But this problem has occurred in real life. The Y2K problem is a live example of that. The software was not expected that it should last long till the year 2000. But it lasted, which resulted as a Y2K problem. Also the legacy systems are developed with the technology that is in use or popular at that span of time period. But the technology for developing systems changes after some span of time. Thus, maintaining systems that were developed with older technologies with the newer technologies seems to be difficult.

Mostly, in legacy systems the trends of maintaining documentation is very less. Which leads to the problems of understanding programs, designs, specifications and maintaining such systems is more than difficult.

4.5 Lack of Adequate Documentation

The absence of adequate documentation also becomes one of the factors that cause problems while maintaining reusable components. The reusable components can be misunderstood or there can be a misconception about the components in the absence of documentation. Documentation can also be a source of reflecting changes, but if it does not exist then the changes cannot be reflected to make the task of maintainers more difficult.

A lack of documentation makes it more difficult for the maintainers to understand the reusable components and also there is a probability that the changes made to the components are left undocumented.

5. Alternatives for Maintenance of Reusable Software Components

There may exist various alternatives as a solution to the maintenance problems of reusable software components. The study discusses some of the major alternatives to the maintenance problems in this section. These are:

5.1 Notification About Changes

Managing and controlling changes to the reusable components in a repository can play a vital role for the success of a software product. Notification about changes can be one of the alternatives to change control of reusable components. Whenever there is any of the modification in reusable components, the changes should be notified by repository manager to different stakeholders of the components. These stakeholders can be the reusers, maintainers, configuration managers, and the remote users as well. All the stakeholders should be notified that the reusable component has been changed. It should also be notified that what changes are made, the type of changes made, nature of changes made, and the size of changes made. It is the responsibility of the repository manager to control the changes in components. It should keep all the information related to reusable components such as log of change, log of reuse, and log of maintenance issues. Changes in environment of the existing system should also be notified to the maintainers. Uncontrolled changes in reusable components which are being used in multiple projects can prove to be dangerous for these projects. So, the changes should always be controlled by established procedures and methods.

5.2 Integrated Environment of Reuse Repository with SCM

An integrated process model of reuse processes and maintenance processes in SCM environment can be one of the alternatives for version control problem. It states
that the processes of reuse and the processes of mainte-
nance are integrated with each other in such a manner
that they can be used actively in configuration manage-
ment environment to manage the component repository. While managing the component repository the number of versions of a component can be controlled. The inte-
grated environment provides an adequate tool support
for managing number of versions. If the reuse processes
and maintenance processes are used separately in order
to manage the versions then there is possibility that the
configuration manager can skip any of the changes in
any of the versions. So, if they are used in an integrated
environment then the configuration manager should be
able to track the changes in number of versions of com-
ponents. A version history of reusable components can
be maintained to track the number of versions for a single
component. The version history can also be accompanied
by the number of changes in a particular version thus
keeping track of changes in version of reusable compo-
nents at a single place.

5.3 Reverse Engineering as Tracing Tool
Reverse engineering can be used as a tracing tool to trace
the reusable components. Maintainers while maintain-
ing components can engineer in a reverse direction of
the component functionality to trace the related informa-
tion about the components. Reusable components can be
reverse engineered to trace out its specification details,
design details and other related information. Reverse
engineering process can be implemented in two ways.
In one way, all the related information of the component
which is obtained after reverse engineering is provided
to the maintainer who can locate the errors and defects
in order to maintain the components. In the other way,
the related information which is obtained after reverse
engineering is fed into a re-engineering process to reconsti-
trute the old system.

5.4 Upgrading Through Re-Engineering
The problem of maintaining outdated legacy systems
can be handled with the help of re-engineering process.
In a re-engineering process the system is reconstituted.
While re-engineering systems their design, architectural,
and data related details are extracted which are then fed
into a re-engineering process while constituting the sys-
tem in order to upgrade the system. After upgradation,
the systems can be constituted in such way that they can
be maintained easily and efficiently. While upgrading the
systems a proper documentation can be created so that
the systems can be traced in the future as well. Also the
systems should be upgraded to the latest technologies so
that there should have less maintenance problems.

5.5 Changes Should be Reflected in
Documents
A proper documentation can resolve the issues caused due
to the absence of documentation of components. If there
exist the components with adequate documentation, then
the task of maintainers can be easier. The components can
be understood easily with the help of documents. Also,
the changes made to the components can be reflected in
the documents to track the history of changes. A history
of maintenance of components can also be tracked from
the documents. Thus, it is advisable to maintain docu-
mentation related to reusable components.

6. Tools that can be Helpful while
Maintaining Reusable Components
A discussion about the tools that can prove to be useful
one while maintaining reusable components is conducted
in this section. It listed out the points of discussion as fol-
lows:

- Reverse engineering and re-engineering are the
  prominent tools that can be helpful while mainta-
ing reusable components.
- CASE tools such as design tools, configuration
  management tools, and sometimes testing tools
  can help the people in performing their tasks.
- In most cases, the configuration management
  tools are useful for the maintainers to maintain
  the reusable components.
- IDE’s software with pictures for the purpose of
  software designing is helpful.
- A tool for configuration management named as
  PROCASE’s SMART System.
- HP’s Softbench or Sun’s ToolTalk for examining
  integration framework.

7. Conclusion
The study identifies a number of problems that can cause
a threat to the maintenance of reusable software compo-
nents. The issues identified are change control, version
Concerns in Maintaining Reusable Software Components and the Possible Solutions

control, traceability of code, lack of documentation, and the obsolete legacy systems. It then analyses these problems and the reasons due to which the issue arises. It also establishes the relationship between the maintenance and reusable software components. While establishing relationship it states that maintenance is related to reusable components through reverse engineering and re-engineering. The study figure out the number of alternatives that can be useful to resolve the stated issues. The alternatives figured out are notification to changes, adequate documentation, integrated environment of reuse repository with SCM, reverse engineering as a tracing tool, and re-engineering as an upgrading tool. Tools and CASE tools that can be helpful for the maintenance of reusable components are also listed.

8. References

1. Software Engineering Standards Committee of the Institute of Electrical and Electronics Engineers (IEEE) Computer Society. IEEE Standard for Software Maintenance [Internet]. 1998 [updated 2015 Sep 8; cited 1998 Jun 25]. Available from: http://www.cs.uah.edu/~rcoleman/CS499/CourseTopics/IEEE_Std_1219-1998.pdf.
2. http://dictionary.ieee.org/index/s-14.html. Date accessed: 09/09/2015.
3. Karanja D. Equipment commissioning checklist [Internet]. 2015 [cited 2015 Sep 8]. Available from: http://www.ehow.com/about_6643440_software-maintenance-definition.html.
4. Gill NS. Software maintenance and maintenance costs. In Software Engineering, Khanna Book Publishing Company; 2009. p. 490–5.
5. Chu WC, Lu C-W, Shiu C-P, He X. Pattern-based software reengineering: a case study. Journal of Software Maintenance: Research and Practice. 2000 Mar; 12(2):121–41. Crossref.
6. A Process Model of Maintenance with Reuse: An Investigation and an Implementation [PhD thesis]. UK, University of Durham; 1997 Dec 26. p. 1–193. http://theses.dur.ac.uk/4724/1/4724_2193. PDF?UkUDh:CyT.
7. Voas J. Upgrading software maintenance for components [Internet]. 2015 [cited 2015 Sep 9]. Available from: https://www.cigital.com/papers/download/maintain.pdf.
8. Murta L, Oliveira H, Dantas C, Lopes LG, Werner C. Towards component-based software maintenance via software configuration management techniques [Internet]. 2015 [cited 2015 Sep 8]. Available from: http://reuse.cos.ufrj.br/prometeus/publicacoes/cbsm.pdf.
9. Al-Badareen A, Selamat MH, Jabar MA. Reusable software component lifecycle. International Journal of Computers. 2011; 5(2):191–9.
10. Imeri F, Antovski L. An analytical view on the software reuse. In ICT Innovations Web Proceedings; 2012 Jan. p. 213–22.
11. Pressman RS. Software engineering: a practitioners approach. 3rd edition, McGraw-Hill, NY; 1992. p. 668–83.
12. Sharma A, Kumar R, Grover P. Managing component-based systems with reusable components. International Journal of Computer Science and Security. 2007; 1(2):60–5.
13. Schach SR, Tomer A. A maintenance-oriented approach to software construction. Journal of Software Maintenance: Research and Practice. 2000 Jan; 12(1):25–45. Crossref.
14. Manna M. Maintenance burden begging for a remedy. Datamation. 1993 Apr 1; 13(6):53–63.
15. English Oxford Living Dictionaries. Version [Internet]. 2015 [cited 2015 Sep 2]. Available from: http://www.oxford-dictionaries.com/definition/english/version.
16. Wikipedia. Version [Internet]. 2015 [updated 2016 Dec 22; cited 2015 Sep 2]. Available from: https://en.wikipedia.org/wiki/Version.
17. Sharon D, Bell R. Tools that bind creating integrated environments. Institute of Electrical and Electronics Engineers (IEEE) Software. 1995 Mar; 12(2):76–85. Crossref.
18. Bakshi A, Bawa S. Development of a software repository for the precise search and exact retrieval of the components. International Journal of Advanced Research in Computer Science and Software Engineering. 2013 Aug; 3(8):1116–26.
19. Pressman RS. Software engineering-a practitioners approach. Software Configuration Management; 2010. p. 584–612.
20. Britcher RN. Re-engineering software: a case study. IBM Systems Journal. 1990; 29(4):551–67. Crossref.
21. Biggerstaff TJ. Design recovery for maintenance and reuse. Computer. 1989 Jul; 22(7):36–49. Crossref.
22. Dart S, Christie AM, Brown AW. A case study in software maintenance. Technical Report; 1993 Jun. p. 1–50. http://www.sei.cmu.edu/reports/93tr008.pdf.