International Journal of Computer Applications (0975 – 8887)
Volume 131 – No.2, December2015

Reusability Types and Reuse Metrics: A Survey

Aditi Dubey
Department of CSE,
Lovely Professional Univers
144411, Punjab, India

Harleen Kaur
Department of CSE,
Lovely Professional Univers
144411, Punjab, India

ABSTRACT

This paper focuses on the reusability of software with types of reuse and metrics of reusability. From the word itself "Software Reuse", it is easily understandable that we are reusing the artifacts of software more than once. Software artifacts are some components of the software system that are used in software development life cycle. Implicit artifacts are – design, code, test plans, documentation and some explicit artifacts – component selections, measurement and maintenance costs. All of these artifacts are reused in different systems but for reusing any artifacts, we require knowledge of each and every domain.

Keywords

Reuse, Metrics

1. INTRODUCTION

In today's world we are surrounded by plenty of software, from developing it to using it. Everyone wants the quality and performance of the software to be the best, without any flaw. For measuring the software's quality and its performance we have software metrics.

The software's productivity and quality can be improved by software reusability. Software reuse is just a reapplication of artifacts of same objects from one application to another. Every Developer builds the software from the scratch, but at the time of software crisis and due to the wastage of time by writing the same code again and again, we reuse the same component. Software Reuse is also the solution that avoids the repeated labor in the software development.

At the time of software crisis, reusability became too expensive and its development was not monitored by the managers, at that time software reuse reduced the overload. Reuse is necessary as the cost used in the development and maintenance of the software is reduced and also improves the quality of the software. We do evaluation and revisment of Software because reusing the software again and again will improve the quality. Reuse of components doesn't mean that only reusing the code from one application to another, it means reusing the design, architecture and other components of the application as well [4].This paper is divided into IV sections. Section II explains the characteristics of software reuse, section III explains Types of reuse, Section IV explains reuse metrics and last section V tells the conclusion and future scope.

2. CHARACTERISTICS OF SOFTWARE REUSE

Some useful characteristic that make software easily reusable or characteristic that should be kept in mind before reusing the software components in another application are[1]-

- Modularity- It refers to the breaking down of the software into pieces. Small parts are always easier to reuse than larger ones. Also each module can be treated separately according to their specific functionalities.
- Loose Coupling- It explains that dependencies between the services should be less. Two modules which are joined together to perform an action should have low coupling (dependencies) between them so as to reuse the modules [10].
- High cohesion- In software development we have different modules. So keeping the same related components together is known as high cohesion. We can access the module later for reusability because it has no dependency on other modules.
- Separation of Concern- This property explains how computer programs are divided into distinct sections, such that each section addresses a separate concern.

Why software reusability was not implemented across the world?

This is because developers or project managers did not know what to reuse and the way of reusing the components. Even nowadays, in some parts of the world the development of a software follows the Software Development Life Cycle where System design is a major part but reusability is not. When system reusability has some major benefits when included in the development life cycle, developers should start considering these aspects and implement it from the very beginning of the development phase as it will be very helpful in the testing and maintenance phase of the system components.

Reuse can be divided into many parts which covers almost all

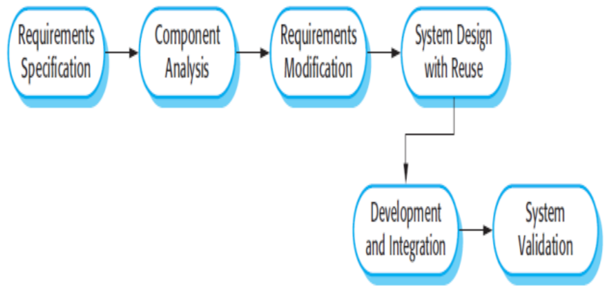
Requirements Specification
Component Analysis
Requirements Modification
System Design with Reuse
Development and Integration
System Validation

Figure 1. Software development life cycle.

3. TYPES OF REUSE

- Development scope.
- Modification.
- Approach reuse.
- Domain scope.
- Management scope.

1) Development Scope

Development scope tells us whether the reusable components came from internal scope or External scope[5].

- **Internal Scope** Internal Source means to use the components of the application in the same application like, reusing of the code from one module of the software into another module.
- **External Scope-:** External Source means when we outsource any component, or when we reuse the code, architecture, and design from other application into our application.

2) Modification

Modification reuse tells us how much reusable assets need to be changed for our own purpose. We have 2 types of modification in Reuse

- **Black Box-:** In Black Box, we use the components in another application or the same application without any change in the code, design or architecture. This approach guarantees higher quality, and reliability but it is expensive to create black boxes. It just reduces the exhaustive testing.
- **White Box-:** In White Box, sometimes customers are not satisfied with the reuse components and it cannot be used until properly modified according to customer’s need. Study of Domain before reusing is very important.

3) Approach Reuse

It tells different technical methods through which we can implement reuse. We have 3 methods of approach reuse-

- **In-the-small-:** It means reuse of small pieces of source code, such as- classes, subroutines and packages.
  Issues faced during reuse in-the-small are-
  1. We have insufficient means to manage the whole process of software life cycle because having the knowledge of only source code is not enough for reuse.
  2. Building a big system by reusing the code only leaves behind lots of work to be done in the whole system, because reusing small codes do not make a big system.
- **In-the-large-:** It means we reuse large grain components such as- subsystem, reuse of the design, architecture.

  In this we have 3 major techniques [3]-

  1. **Reverse Engineering-:** In this, we analyze the entire system to identify all the components of the system and their interrelationship so as to know which components can be reused, the way of reusing them, and to create a representation of the system in another form or at higher level of abstraction.
  2. **Reengineering->** It aims at evolving old system or **building** a new software system out of an existing one by utilizing the information provided by the reverse engineering.
  3. **Reuse supported forward engineering->**In this engineering, process goes from requirement to the implementation process.

  Reverse engineering and Reengineering are applied to poorly designed and incomplete products without any structure supporting the reusability. To avoid this process, reusability of the components at a pervious stage of the software development is the best solution.

  Different ways to perform Reuse in-the-large are-:

  1. In small reuse we only reuse the code but reuse of information is more precious than code components and the cost of design is also greater than that of code.
  2. Reuse of any large grain component requires proper knowledge. Design information is efficient only when we have domain knowledge of that part.
  3. Additional cost of building up a system from small code components can be reused if software architecture is reused.
  4. Software Architecture is the basis of organizing both code components and the design information.
- **Compositional Reuse :** Compositional reuse is the use of existing components as building blocks for new system. It is mode for software architecture which compose existed architecture design resource to form larger architecture. It includes well-established standard interface and library system. It mainly focuses on source code reuse.

4) Domain Scope

Domain scope tells whether reuse occur with family of system or between families of system and it is a process of identifying, catching and organizing the reuse information of alike object and operation inside the system in particular domain. Until and unless we don’t have domain knowledge we can’t reuse that part in another application. Some of the domain scopes [6] ->

- **Vertical Reuse-:** Vertical Reuse is the reuse done within the same domain or application area. Its main motive is to generate generic models (generalization of conventional data models) for families of the system that can be used as a template for assembling new system. In this approach we mainly deal with identification and development of domain.
- **Horizontal Reuse-:** In this, we create generic model and here we use the parts of that generic models in different application.

5) Management Reuse

Management is very major part when we are reusing any other components because we have to pay proper attention to that part. Degree to which reuse is done systematically comes under this. To maintain proper reusability we divide it into 2 parts –

- **Systematic Reuse**- Systematic Reuse is the planned reuse in which specific components are identified as a reusable one at a specified location. In this we do normal practice of reuse and procedures for reuse have been defined [7].
- **Ad-hoc Reuse**- It is also called opportunistic reuse, in this developers take the decision to find the reusable elements, retrieve it and then reuse it. Procedures for reuse doesn’t exist.

Diagrammatical explanation of how much reuse is necessary and how it reduces our efforts.

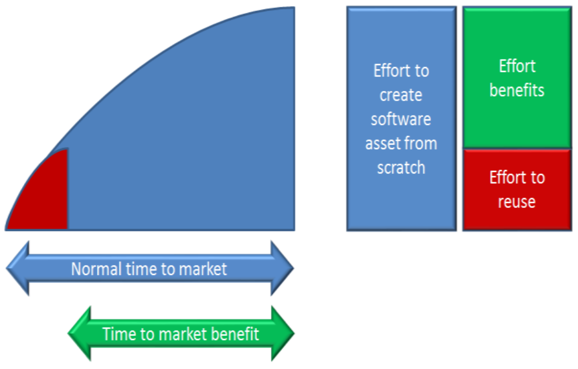

Figure 2 Efforts required in reuseSome important analyses have been done on all the types of reuse to get a structured idea on it.

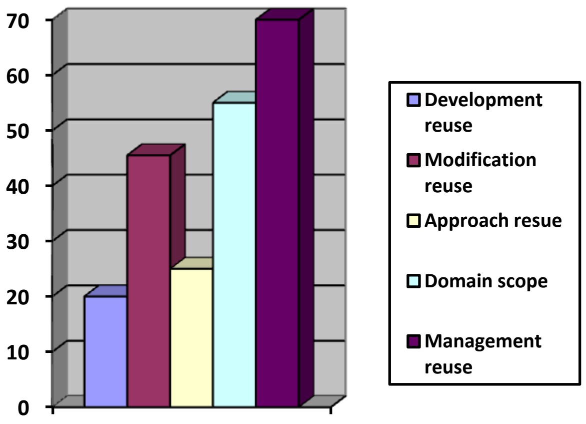

Figure 3

From this histogram we see that management reuse is the most important and it is more demanding. At this time because we basically work on the systematic reuse. After that domain scope come into play because reusability is done after seeing the domain. Domain knowledge is very necessary for reusing its parts. But all types of reuse has its own importance and according to the scenario reusability is done.

4. REUSE METRIC

As an organization implements the software reuse program to improve productivity and quality. Many times for complex IT organizations and companies it is very difficult to measure the software's productivity and ensure the quality of the software. So, we have software metrics to measure the application specification to provide objective and any repeatable data for making an improvement in the suitable areas.

• Types of metrics.

For making any software reusable, at the time of creation only we have to think of the reusable assets so that design of that software is suitable for major change in the requirements. Once any change is made to any phase of that software then all the phases of that software should adapt to that changes. This is perfect system for the reuse [8].

We have 6 types of metrics to measure the quality-

- Reuse Cost- Benefits
- Maturity Assessment
- Amount of reuse
- Reuse Library metric
- Failure mode
- Reusability Assessment

1. *Reuse Cost-Benefits*

This technique involves adding up the benefits of the course of action we perform and compare these with the cost associated with it. One example- suppose we make a health application in which we create some new modules and reuse some previous modules. If the integrated application is cost efficient than the normal application then we will use that application only. I quote this example to explain u that if reusing any module gives u benefit in monetary term then only we will reuse that part. Cost of any reuse program depends upon small and large application domain. If we have small application domain in which only few components are reused then cost associated with it is less and calculation of the cost is also easier [11].

But if we have large application domain which is too large and complex and is not made or understood by single person it requires a team in every module, then its components which are reused are costly. Cost is checked in every phase of the Software Development Cycle, 1st phase is to check the Feasibility of the project whether it gives the appropriate benefits or not.

Use of Cost Benefit Tool

1. Brainstorm costs and Benefits->In this we consider the overall project requirements, its needs and the resources e.g.-the labor required in the project, the type of hardware and software required etc. we should consider all the cost associated with the project and the benefits that we achieved with that resources.

2. Assign a monetary value to Cost->This includes cost of physical resources needed, as well as cost of human effort that is involved in every phase of project. Sometime if the technology is new or the project is bigger than the training cost is also considered.

3. Assign a monetary value to the Benefits->In this we quantify in monetary terms the benefits arising out of the projects.so that if the quantity is not according to the cost we invested in the project then cost-benefit analysis gives no benefit. In software world identification and quantification of benefits is very difficult and time consuming.

4. Compare Cost and Benefits->In this we compare the value of our costs to the value of our benefits and then use this analysis to decide our course of action. Calculate the total costs and the total benefits and then compare the two of them to determine whether our benefits outweigh our costs or not. In many projects we can't see the benefits after the completion of the project it takes time so to we use the term Payback period- how long would it take to reach to break point means the point at which benefits have started repaying the cost.
This is a very good tool to decide whether to pursue a project or not.

2. *Maturity assessment*

In this we have many programs from them we categorize the reuse one and then see how advanced they are in implementing the systematic reuse. It is a self-evaluation type tool in which it checks where our project stands and where it needs to be so that we can bring it to that level. Measures through which we know the performance is performance indicators, with 3 key area of assessments-

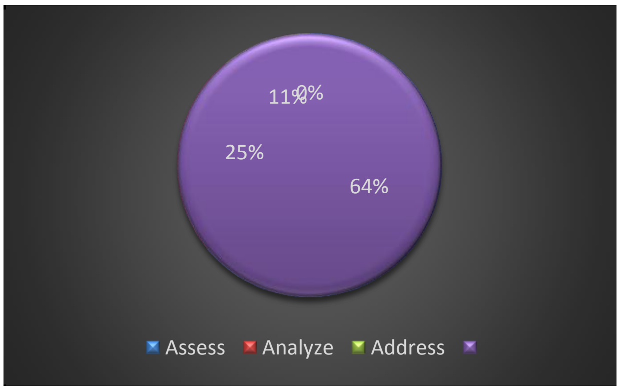

Figure 4

From the 3 key area of assessment *Assess* is the important one. In this, we have important parameters that need to be considered while creating a project. We create a chart and provide each parameter a score which defines the importance of the parameter and also helps the leader of different team to see the current state of the project through graphs. After that comes another *A-Analyze* in this leader create score card and give it to

head who analyzes the parameter.

What's the need of analyzing the data? Because we have many types of parameters, so we analyze some important parameter on which we have to actually start working. *3rd A is-Address* while building up the software we came across some weakness so leaders addresses the weakness and then start working on it. How to improve weakness? Leaders announce loudly the weakness to all the team mates and then all the teammate discuss about that weakness and give solution to improve it. The one which is cost-benefits and provide benefit on time is applied.

3. Amount of reuse

It tells us how much or what percentage of data we have to reuse to make our new system more efficient. When Line of code is reused to another system or same system then we have to check the complexity of that code also and also we see that from 1000 LOC how many lines have been used. Some important subcategories of amount of reuse are [9] –

a) Reuse level- A system or a program composed of parts at different level of abstraction e.g. packages, functions, Line of Code (LOC). One example to explain this- let's have 1 program to calculate the attendance of students in college and another to make a time table for students. So firstly programmer will make program of making time table and when he/she goes to next module that is to create program for attendance he/she wants to calculate the hours per day a lecture taken. So in this one item or only that hour part will be fetched from the time table module. Not whole module of time table is used, some level is only reused.

b) Reuse percentage- to calculate how many lines of code the program has reused from different module then we use percent (%).

$$\% \text{ reused} = \frac{\text{number of reused LOC}}{\text{total number of LOC}}$$

for e.g. In Asp.net when we do html coding then some percent of code is reused in all pages of the website, it means that some line of code is reused.

c) Reuse size- when we have to choose the lines of code then not always the lines decide the size of the program, mainly we have to know the complexity of that program code. Sometime the code of 4 line have highest complexity and lines of 20 code have less complexity so it's good to use code with less complexity, to make the system effective.

4. Failure modes

It analyzes the failure modes, due to which the reuse in the organization is not possible. That failure modes are used to evaluate the quality of the system, and for the improvement strategy of systematic reuse. There are many factors that affect our reuse strategy to be successful. We have to know that which reuse program we should use to recover that failure modes, but for knowing this our organization must answer the question that why reuse is not taking place in the organization?

5. Reusability Assessment

This module tells that whether the component or the artifact is reusable or not. One important question come into my mind is- Is there any measurable attribute that indicates its reusability? If we know this then we can easily know that whether the component is reusable or not. Components are used from existing system to the new ones so they are more reliable than the components that are made from the scratch. Before using the components of one system to another are they tested? The answer is yes because after testing only they are attached to new system. After so many test we can't reuse the component due to lack of documentation, lack of experience for reuse.

1. **Reuse Library metric** Library is a repository which is used to store data and to have effective search because once data is in correct order the searching process get effective. Reuse Library means where we keep our reusable assets with proper management. In reuse library we can search our component which is needed for reusability and in library every item is arranged according to their properties so search becomes effective and time saving. From where did we get the reusable assets? It can come from the existing system through reengineering or we can purchase them or sometime many programmers create the code for reusability means the design of that component is reusable itself. These reusable assets are then certified because to keep any product in the library we should know its property to classify them properly.

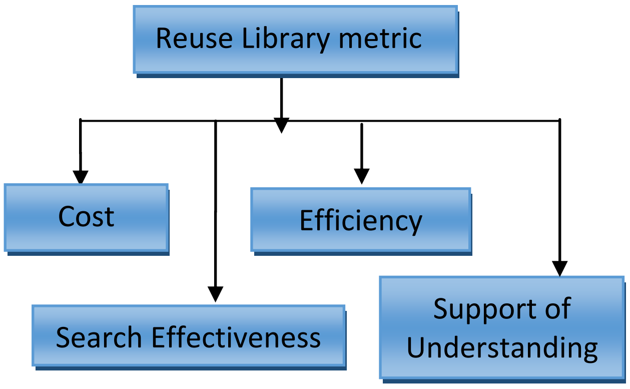

Figure.5 Types of reuse library metric

1. **Cost**- Indexing cost is included in classification like how to classify the elements into reusable parts, cost require for maintaining that part and then for updating it, because new elements keep on adding so updation is very necessary. Human effort is required to maintain the process.
2. **Search Effectiveness** – After classifying the components, how easily and the user is able to get its components. This method saves our time because we can easily assess our assets.
3. **Support of Understanding**- To get the component easily is not what the engineer wants they should properly understand the meaning of that reusable component, so that they can use them more effectively.
4. **Efficiency**- While storing the data how much memory has been used tells the library efficiency and how to calculate the memory usage? It can be measured by the number of bytes that are used to store the data. Efficiency also depends on how faster the data has been retrieved and it is calculated by the time it takes to search a particular query from a database.

5. CONCLUSION AND FUTURE SCOPE

In this paper, we presented a survey of reusability and types of reuse that reduces the development time and effort and the most important cost of the application, because we don't have to make all the parts of that application from the scratch. Some parts are reused. A reuse program should be always be planned, done carefully and should be in systematic manner to give the higher payoff. Software reuse strategies are implemented in an organization to improve the quality and productivity of the software. To measure the quality and the productivity of the software metrics are used. Metrics is a quantitative indicator of an attribute.For reusability, domain knowledge is very important. Without proper knowledge of what to use and where to use the application cannot give us qualitative product with efficient time. So Domain engineering plays an important role in reusability. Domain knowledge is the key concept of systematic reuse.

6. REFERENCES

[1] T.Karthikeyan, J.Geetha, "A Study and Crtical Survey on Service Reusability Metrics",I.J. Information Technology and Computer science",2012,5,25-31.

[2] Arun Sharma, Rajesh Kumar & P.S Grover. "A Critical Survey of Reusability Aspects for Component Based Systems", World Academy for Science, Engineering and Technology 33, 2007.

[3] Haikuan Li and Jan van Katwijk. "Issues Concerning Software Reuse-in-t he-Large", Delft University of Technology Faculty of Mathematics and Computer ScienceJulianalaan 132, Delft, The Netherlands.

[4] Yong-liu, Aiguang-yang, "Research and Application of Software-reuse", College of Information Science and Technology, Qingdao University of Science and Technology, Qingdao, Shandong, China.

[5] William Frakes Carol Terry, "Software Reuse and Reusability Metrics and Models", Virginia TechComputer Science Department2990 Telestar Ct. and Applied Expertise, Inc.1925 N. Lynn St. Suite 802Arlington, VA 22209.

[6] Swati Thakral, Shraddha Sagar and Vinay, "Reusability in Component Based Software Development - A Review", Galgotias University, Gr. Noida, India.

[7] James M.Bieman, "Deriving Measures of software reuse in Object Oriented System", Department of Computer Science Colorado State University.

[8] Jeffrey S. Poulin, "Measuring Software Reusability", Loral Federal Systems–Owego.

[9] Jorge Cláudio Cordeiro Pires Mascena, "A Comparative Study on Software Reuse Metrics and Economic Models from a Traceability Perspective", Eduardo Santana de Almeida, Silvio Romero de Lemos Meira Federal University of Pernambuco.

[10] Suchita Yadav, Dr. Pradeep Tomar, Sachin Kumar, " Metrics Suite for Accessing the Reusability of Component-Based Software", School of ICT Gautam Buddha University, Greater Noida, India.

[11] William, B. F. and K. Kyo (2005). "Software Reuse Research: Status and Future." IEEE Trans. Softw. Eng. 31(7): 529-536.