ISSN:0975-9646

Component Based Software Reusability and Framing Standards for Components to be Reused

Anshul Kalia[1], Sumesh Sood[2]

[1]*Research Scholar, Department of Research, Innovation & Consultancy, Punjab Technical University, Kapurthala – 144601, India*

[2]*Head of Department (Computer Applications), Swami Satyanand College of Management & Technology, Amritsar – 143001, India*

***Abstract* – Software components play an important role in achieving reusability. Reusability is in fact an advancement of component based software engineering. A component acts as a basic entity while adopting a reuse approach. Reusability puts a positive impact on reduction of software cost and improvement of software quality. In order to yield the benefits of component reusability it is required to have complete understanding of the activities that are performed while developing software. A component must be developed using some standards so that it can be reused. Certain standards are provided to make a component reusable. A due attention should also be given to organizational issues concerning the software reuse such as infrastructure required, legal issues, incentives etc. Reuse maturity model helps in measuring the level of reusability in an organization. Cost related issues also need to be addressed carefully and effectively. The reason is that monetary issues are a major concern while adopting the reuse approach.**

***Keywords* – Component based software engineering, Reuse maturity level, Software reusability, Component standards.**

I. INTRODUCTION

Computer software has emerged as a driving force in the global IT world. It is like an engine that drives us to a decision or a solution through scientific analysis and investigation. A huge amount of expansion in the use of software has been seen. A software is embedded in systems of all kinds: medical, telecommunication, office management, military, transportation, industry operations, administration, research, entertainment etc ……. the list is almost endless [11]. A software is used by almost everyone in the world either directly or indirectly because most of the daily life activities are affected by it. The softwares have become so user friendly and easy to use that in a manner they have changed the trends of the industry. The increasing number of software users are non- experts. As a consequence of this there is a change of demands on software.

With the wider area of software reach and change in demands of users, developing a large and complex system that should be able to meet time & delivery deadlines, budget, and quality requirements is not an easy task. But all these issues can be resolved by using the concept of software reuse. It is an old as well as a new concept [11]. Reuse of a software component can yield better returns in the form of monetary, reliability, quality, time – to – market etc.

A marketplace for software components is emerging. Component based development can be addressed as a subset and also as an extension of software engineering practices. The idea behind the component based software engineering is to satisfy the development needs of the system by taking it as an assembly of subsystems of a system. The subsystems (components) involved in developing a system must be able to be treated as a reusable entities [4]. By building systems out of carefully designed, pre-tested components, one will save the cost of designing, writing and testing new code. The idea of software reuse seems to be simple but in practice, there are several aspects like organizational, economical, technical and operational that needs to be overcome.

II. LITERATURE REVIEW

McIlroy [1968] first envisioned Software reuse, at a NATO Software Engineering Conference, where he predicted that mass-produced components would end the software crisis [7]. The final objective was very clear: to make something once and to reuse it several times.

Frakes and Terry [1996] – was first person to propose metric and models on software reuse. He suggested models based on cost benefits, assessing the maturity level, reuse library metrics [9].

Kim [2005] takes – on the issue of component based software reuse. It discusses the difficulties in realizing the component based software reuse and to discuss the pre – conditions required to meet before practicing software reuse on a wide scale in a formalized manner [8].

Sharma et. al [2007] discusses about managing component – based systems with reusable components. It discusses the reusability concepts for components based systems and to explore the reusability metrics to measure reusability directly or indirectly [13].

Anas al – badareen [2011] proposed a framework that contains the extraction, adoption and storage of reusable software components [1].

Gupta and Kumar [2013] conducted a study about reusable software component retrieval system. The study discusses the techniques for storage and retrieval of software components which can be reused [6].

III. COMPONENT BASED SOFTWARE ENGINEERING AND REUSABILITY

In component based software engineering approach, the emphasis on components is more. A component can be treated or defined as an independent system that accomplishes a specific task. A component also can be composed of several other components. More specifically software components are prebuilt items that act as a building block in a software to perform specific functions. These components can communicate with each other using standard interface. After the strengthening of component based software engineering practices, the idea of software reuse has evolved more significantly. In fact component based software engineering is a process that emphasized the designing and construction of software using reusable software components [11].

Reusability shifts focus from programming software to composing software system [11]. There lies many advantages of component reuse like reduced development time and cost, the quality and productivity will also be increased as the pre tested components are being used. But the developers could not yield much benefit from it. There may be several reasons for that but the lack of proper definition for the components which is to be reused can be the one reason. The non availability of an established definition for a component can also create confusion and misconception among the developers. It can be explained with an example: you purchase a stereo system and bring it home. We can fit many components like speakers, earphones etc. into it. Each of these components has been designed to fit a specific architectural style, the connections between components are standardized, and communication protocols have been pre-established. Assembly of such components is easy than to build a system from hundred of discrete parts. This is what we need to achieve [11]. A component must be designed in such a way that it should be able to

a) Integrate and communicate with other components in the same system easily.
b) Integrate and communicate with other components outside the system but within same domain.
c) Also it should integrate and communicate with applications outside the domain.

There are several questions about component reusability which are still unanswered, such as, can a large system be built by composing the reusable components, will it be possible to find the components that already exists, how the library of components can be created and made accessible [11], how the incentives will flow to the developer of a component. There is a need for new methodologies and tool support that favours the design, development and maintenance of reusable software components.

IV. ORGANIZATIONAL ISSUES ASSOCIATED WITH SOFTWARE REUSE

Today the organizations are always under pressure for the timely delivery of software product to market, to reduce the development and maintenance cost and to increase the quality of the product. To meet all these deadlines and requirements, industry is shifting largely to software reuse [5]. Software reuse can play a significant role in achieving market deadlines, cost saving, increasing productivity and quality assurance. The two steps have been prescribed to start with reuse process in the organization.

A. Initiation of Reuse Process

The first step that is to be initiated by a software development organization is the assessment of software reuse [4]. Software reuse assessment is performed in an organization to measure the potential for practicing reuse. It is vital in determining that whether the organization is ready to take on the reuse program or not, also it defines for the organization to where to focus its reuse efforts in order to gain maximum benefits from reuse practices. The resultant information obtained from reuse assessment can be used as a basis for defining organization's reuse goals, strategies for reuse adoption, the domains in which to practice reuse and the reuse implementation plan [4].

Reuse assessment helps in successfully introducing reuse into software development organizations. The objectives of reuse assessment are:

- To analyze and evaluate the organizations current reuse policy and implementation of that policy in current software projects.
- To find the organizations reuse goals, identification of elements of reuse program to achieve these goals.
- To determine the area of domain towards which the reuse efforts can be directed to obtain maximum benefit.
- It provides the course of actions to be taken to implement the reuse policy.

Institutionalizing the reuse practices in a large organization is a complex task. For successful implementation of reuse practices in an organization, the organization needs to introspect that how ready, willing it is to adopt the reuse development approach, what steps it needs to take to prepare it to achieve reuse goals. The assessment include following issues [5]:

- Identification of personnel support required for implementation of reuse program.
- Establishment of infrastructure for reuse i.e. reuse incentives, reuse corporate policy.
- Business value of a reuse program.
- Role of management in reuse program.

B. Measure of Reuse Activities in an Organization

Reuse maturity is a measure which is used to determine the effectiveness of reuse activities in an organization [12]. By evaluating the maturity of reuse in the organization we determine the level of activities, and accordingly it can be increased if required.

A five level maturity model has been proposed by Koltun and Hudson [12]. In this model, the discrimination among the levels is created considering motivation, planning for reuse, breadth of reuse, responsibility of making reuse happen, process by which reuse is leveraged, reuse inventory, technical support, metrics and legal considerations. The five levels in model are initial, monitored, coordinated, planned and ingrained. Table 1 gives the overview of various characteristics of maturity model [12].

TABLE 1
CHARACTERISTICS OF REUSE MATURITY LEVELS

Level	Characteristics
1. Initial	Short term thinking Reuse costs are feared Resistance to reuse Individualized reuse i.e. uncoordinated, unmonitored
2. Monitored	Managerial awareness Reuse costs are known Little active promotion of reuse Individual achievements
3. Coordinated	Organizational responsibility Domain analyses for product line Reuse tactics Payoff of reuse is known Component standardization
4. Planned	Life cycle view of reuse Reports of reuse costs and savings Reuse is supported and encouraged Reuse across all functional areas
5. Ingrained	Corporation wide view Reuse is regular business Domain analyses across all product lines Corporation wide definitions, guidelines, standards

The reuse maturity model can be used as an assessment tool that can be used to measure an organizations maturity of reuse processes against a set of characteristics that are specified at each of the maturity level. The more mature are the reuse processes, the more likely an organization is to produce software products on time, on budget and on to the specified quality. The reuse maturity model can also be used by the organizations for raising the maturity of their reuse processes and to determine the practices that will actually increase the maturity of their currently running reuse processes [5].

V. STANDARDIZING COMPONENTS TO BE REUSED

Reusing of a component requires a set of standards in order to clearly define the reusable components. Standards provide the framework for the components to be developed. The components so developed can fit easily into the software systems that are to be developed using reusable components. Also the developers should be able to understand the components quickly and easily. By using a set of standards a component can be made reusable. The cost of making a component reusable must be taken into account while conducting a cost benefit analysis. The following are the certain points which are useful in standardizing components:

1. For a component to be reused it must have an interface and it should be consistent.
2. The functionality of a component must be defined.
3. The designing of a component must satisfy a specific style of architecture.
4. A component must have industry defined communication protocols that enables it to communicate with other components in same application, outside the application but within same domain, outside the domain.
5. Run time requirements (i.e. memory, storage space), time and speed, network protocols must be defined.
6. For integration with other components a component must have an application programming interface.
7. A component must also be able to define that how it will navigate to a database to create, edit, delete and to perform other operations as instructed by the end user.
8. It also requires data exchange mechanisms that enable users and other applications to interact and transfer data i.e. cut, paste, drag, drop etc.
9. Security features including access controls and authentication protocols should also be defined.
10. It must have the ability to handle errors and exceptions.

The points raised above will surely be helpful in component definition. It will make the process of understanding and integrating components fast. The components that should be developed using above said standards can be adopted easily and quickly into the system to be developed. It should be much like assembling a stereo system as we specified in the example given in section 3. In case of software systems it is the pre – developed components within the prescribed frame of standards that are assembled to generate a system as a whole.

A. Benefits of standardizing components

There can be several benefits of standardizing components. Some of them are listed below:

1. It is useful in achieving forward and backward compatibility of the components with the hardware and software.
2. It will be easy to adopt and integrate the standardized components into the system to be developed.
3. Understanding and utilizing such components will be cheaper and less time consuming.

4. There will be an increased reliability and quality of the system under development using standardized components.
5. Also the maintenance of such components and systems developed using such components will be cheaper and less time consuming.

VI. COST BENEFIT ANALYSIS

Before directly jumping to the idea of reuse the organizations must perform the cost benefit analysis, which will be helpful for them to determine whether the reuse approach will yield significant returns on the investment made on it. If the cost of reuse exceeds or equal to the actual development cost, then there is no point of taking reuse path. The cost of software reuse need to be justified with the benefits expected [4]. Only then it becomes feasible to adopt the reuse approach for software development. Cost benefit analysis alone cannot be the criteria for reuse but it can be one of those. The analysis follows an economic model which involves producers, consumers and the distribution chain. While calculating benefits of reuse everyone of these is taken into account.

The net savings can be obtained as below [4]:

$C_{save} = C_s – C_r – C_d$

Where C_s – is the cost of project developed from scratch

C_r – is the overhead costs associated with reuse

C_d – is the actual cost of the software as delivered

The cost of project developed from scratch can be determined by various estimation techniques. Now, the overhead costs associated with software reuse include [4]:

- Training of personnel in design for reuse and design with reuse
- Creation and operation of a reuse repository
- Royalties and license costs of externally acquired components
- Maintenance and updating of reuse components
- Increased documentation to facilitate reuse
- Converting a component into a standard component for reuse

C_d will include project related reuse costs, such as the adaptation and integration of reuse components.

The cost related benefits can further be classified into two:

- The producer, a creator of reusable component
- The consumer, a user of these components for the creation of other software

It is not sure that developing a component for reuse will always be beneficial for the producer, it may or may not be cost effective for the producer but may produce benefits for the user. Producer costs include the cost involved from creation to maintenance of a reuse program [4]. The costs here refer to the whole life cycle cost of a component i.e. analysis, designing, coding, testing, debugging, maintaining [5]. As it will take more time for the producer to make a component reusable which will increase the cost, but on the other hand, it will take less time for user to adapt and integrate which in turn serves as cost effectiveness for a user and a costly affair for the producer.

VII. CONCLUSION

Existing software or a software component can be reused to produce a new system without actually producing it. It can be a composition rather than the production. This paper highlights the issues which are primarily need to be addressed for the start of the reuse process. It also exhibits the importance of organizational issues which may not be given the due attention. A set of standards for components to be reused have been outlined, it will be beneficial in such a way that production, selection, adaptation, integration of components will become easier. Reduction in costs and time – to – market has always been an issue for the software industry. Industries desperately need a shift to software reuse. Thus software reuse will bring the improvements in productivity, quality and reliability of the software.

REFERENCES

[1] Al – Badareen A., Selamat M.H., Jabar M.A., “Reusable Software Component Lifecycle”, International Journal of Computers, 5(2), pp. 191 – 199, 2011.

[2] Crnkovic I., “Component Based Software Engineering – New Challenges in Software Development”, Software Focus, 2(4), pp. 127 – 133, 2001.

[3] Frakes W.; Terry C., “Software Reuse: metrics and models”, ACM Computing Surveys, 28(2), ISSN: 0360-0300, pp. 415 – 435, 1996.

[4] Gill N.S., “Reusability Issues in Component-Based Development”, ACM SIGSOFT Software Engineering Notes, 28(4), ISSN: 0163-5948, pp. 1-4, 2003.

[5] Gill N.S., *Software Engineering*, New Delhi: Khanna Book Publishing Company, ISBN: 978-81906116-3-3, 2009.

[6] Gupta S., Kumar A., “Reusable Software Component Retrieval System”, International Journal of Application or Innovation in Engineering and Management, 2[1], pp. 187 – 194, 2013.

[7] Imeri F.; Antovski L., “An Analytical View on the Software Reuse”, ICT Innovations 2012, Web Proceedings of the 4th ICT – ACT Conference, Ohrid - Macedonia, ISSN: 1857 – 7288, pp. 213 – 222, 2012.

[8] Kim W., “On Issues with Component – Based Software Reuse”, Journal of Object Technology, 4[7], pp. 45 – 50, 2005.

[9] McIlroy M.D., “Mass Produced Software Components”, Software Engineering: Report on a Conference by the NATO Science Committee, Brussels, pp. 138 – 155, 1968.

[10] Morisio M.; Ezran M.; Tully C., “Success and failure factors in software reuse”, IEEE Transactions on Software Engineering, 28(4), ISSN: 0098-5589, pp. 340–357, 2002.

[11] Pressman R.S., *Software Engineering – A Practitioner’s Approach*, 5th ed. New York: McGraw-Hill Inc., ISBN: 0073655783, 2001.

[12] Sametinger J., *Software Engineering with Reusable Components*, Springer-Verlag, New York, USA, ISBN: 3-540-62695-6, 1997.

[13] Sharma A., Kumar R., Grover P., “Managing Component – Based Systems with Reusable Components”, International Journal of Computer Science and Security, 1[2], pp. 60 – 65, 2007.

[14] Wentzel K.D., “Software Reuse – Facts and Myths”, ICSE ’94 Proceedings of the 16th international conference on Software Engineering, ISBN: 0-8186-5855-X, pp. 267-268, 1994.