Volume 8, No. 3, March – April 2017

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

SMALL DATA CHALLENGE: HOW TO USE OTHER PEOPLE'S DATA WHEN YOU DON'T HAVE YOUR OWN - SOFTWARE REUSABILITY

Lakkysetty Nikhil, Praveen Raja, Kamisetty Yeshwanth* and N. Jaya Krishna Naidu
Dept. of SCOPE, VIT university
Vellore, India

***Abstract*:** The concept of using a segment of source code again and again in order to add new functions with minimal or no modifications is known as Reusability. By use of this concept, it localizes code modifications when a change in implementation is required and will eliminate bugs. A code reusability deals with two challenging issues i.e. i) deciding which part of the code to be retained same and ii) deciding which part of the code will be need to be designed from scratch.However, there are numerous approaches such as identifying and qualifying of reusable software components based on coupling, inheritance, external dependency, and polymorphism and reuse index of the component, through newly implemented tool DROOP which stands for Detection of Reusable components in Object Oriented Programming using Java NETBEANS & Java Swings, through multi-layer architecture, through frameworks, through Component Based Software Engineering(CBSE) and various other techniques. The main aim of this paper is to compare and analyze the various reusing techniques in order to find out the best one.

***Keywords*:** Reusability; Components; software; metrics; code; architecture

***Abbreviations*:** CBSE - Component Based Software Engineering; DROOP – Detection of Reusable components in Object Oriented Programming; TCC—Technical Communication Corporation; DIT—Depth of Inheritance, OOS—Object Oriented Software; RLF—Reusability Library Frameworks

INTRODUCTION

As budding programmers and software developers we are often encouraged to reuse our code, some of us are even taught mantras like DRY (Don't Repeat Yourself). But more often than not we start coding from scratch, without even a second thought as to whether we're repeating ourselves. Existing code reuse tools do exist but are often poorly documented and unintuitive to use.

The set of guidelines to help user to judge the quality of the component that is to be reused is known as reusability of code. It is necessary to achieve how much effectively the software component is reusable. To achieve this, the component identification is mandatory. The quality of a reuse system is highly dependent on the quality of the components. There are many ways to reuse code, ranging from the simple, but popular, copy-and-paste to buying off-the shelf software from a vendor and customizing it to specification. This research however is not concerned with these types of code reuse but the reuse of codified algorithms; codified algorithms being methods, functions, subroutines, subprograms or code blocks, depending on what language one's using our approach for identifying and qualifying of reusable software components is based on coupling, inheritance, external dependency, and polymorphism and reuse index of the component.

This paper aims to analyze the source code of these software to identify pieces of code that may form reusable components. Our motivation is that components mined from the analysis of several existing software will be more useful (reusable) for the development of new software than those mined from singular ones.

II. SURVEY ON SMALL DATA CHALLENGE (REUSABILITY OF CODE)

Er.NehaBudhija [1] presented about the reusability, it is the concept of using a segment of source code again and again to add new functionalities with slight or no modifications. They proposed numerous approaches for identifying and qualifying of reusable software components which is based on coupling, inheritance, external dependency, and polymorphism and reuse index of the component. They implemented tool is tool DROOP stands for Detection of Reusable components in Object Oriented Programming using Java NETBEANS & Java Swings. This designed system discovered by them is suitable for reusing components to assist developers by providing them automatic functionality of reusable components searching.

Ankita Mann [2] presented about all the existing metrics on implemented elements and inherited elements to measure quality of design. The proposed cohesion can be increased or decreased depending upon design structure of super class and sub class by including inherited elements. To measure design complexity of software, TCC and LCC are used. LCC aids in assessing the reusability of code which uses the concept of public, private, protected and internal elements require investigation.

Priti Srinivas Sajja [3] proposed a general architecture which is divided into three layers. The first layer is Agent Layer. This layer consists of multiple agents for different activities in a system. For each activity an agent is designed. Agents in the architecture are: Pattern matching, Diagnosing agent etc. These agents, in order to perform their intended activities, they use pre-developed code or tools provided by the Library Layer. And the other is Knowledge layer. Their proposed library layer of architecture includes reusable code to develop software on demand in dynamic manner Automatic Generation of agents using libraries necessitates to generate the code once. Once agents are created, they are

verified by the host or administrator for reliable and guaranteed execution. Once thorough testing is done, the code of agent is entered into the code repository for future use.

Geetika Batra [4] proposed a methodology describing that it contains code repository for the reusability. To support this approach, they used analogy estimation technique for comparison of newer requirements from previous codes. It is necessary to collect historic projects for the reuse of code and design a code repository. This implementation describes that concept of analogy estimation with cosine similarity as search technique of previous code similar to the new required code.

Anas Shatnawi [5] proposed an approachComponent Based Software Engineering (CBSE)which is one of the most important approaches that support software reuse, it proposes an approach to mine reusable components from a set of similar object-oriented software, which were developed in the same domain, ideally by the same developers. Their proposed approach(Romantic) is mainly based on two models: first an object-to-component mapping model, second a quality measurement model to evaluate the quality of components which are mined from object-oriented source code. Based on these two models, mining components from similar software provides more guarantees for the reusability of the mined components rather than depending on single software.

Karabo Selaolo [6] proposed the use of ontologies in tackling these issues, seeing as ontologies inherently encourage reuse. Their research is concerned with the reuse of codified algorithms; being methods, functions, subroutines, subprograms or code blocks, depending on what language one's using. In order to understand common software engineering knowledge and to perform certain types of computations, the software engineering ontology enables the communication between computer systems or software engineersThe ontology cluster is designed for use by agents, both human and software alike, however the relationship goes even further when it comes to software agents. There are many tasks that could be done by agents that will initially be done by simple scripts. Integration with agents was considered out of the scope of this initial building of the system and thus has been left for future visits into this project. In the conclusion of this paper, it is believed that the proposed ontology cluster is a viable alternative to current code reuse tools.

Manoj H.M[7] proposed a model using stochastic based Markov model to find that proposed system can extract significant information of maximized values of code reusability with increasing level of uncertainties of software project methodologies.

Neelamadhab Padhy[8] proposed the new metrics which calculates the reusable codes in the object oriented programme and it is one of the combinations of CK metrics suite. They presented that DIT (Depth of Inheritance) has positive sign on the reusability of the class. If there are more number of methods in the class, then more impact will be on the children class and restrictive the possible of reuse. They concluded that the OOS (Object Oriented System) using the parameterized constructor in C++ programs is more reusable up to some extent. When the larger ethics (values) of proposed Metrics-2 and - 3 are obtained then definitely it gives the negative collision on the reusability. So the negative impact on the reusability of the classes is given by the parameterized constructor.

Priti Srinivas Sajja [9] presented the generic neuro-fuzzy framework which provides advantages such as reusable of code,where there is no need for the user to write codes for the component, the codes can be attached on demand. Now to increase the quality of the system, user may concentrate on analysis and design of the system. He presented that the framework is divided into three layers. In first layer reusable codes such as neural network, fuzzy logic and neuro-fuzzy systems are stored as generic independent objects. The second and third are database and interface layer respectively. Users need not deal with the background code, instead they can deal with the framework.

Dr. Vivek Chaplot [10] that the concept of Reusability is strongly supported by C++. Once a class is created, it can be used by another programmer to suit their requirements. One can create new classes, reusing the properties of the existing classes. This concept of deriving a new class from an old one is called inheritance.

V. Subedha, Dr.S. Sridhar [11] presented on reusability assessment model and different approach in the existing literature which helps the user for mining the suitable component in terms of reusability. When an organization decides to implement software reuse programs to improve productivity and quality of the system development, then they must be able to measure the quality of the reusable software components and identify the most effective reuse strategies. They explained about metrics such as Size Metrics, Coupling metrics, Cohesion metrics and Variability Metrics.

Neha Budhija, Satinder Pal Ahuja [12] presented on empirical study of the software reuse activity by expert designers in the context of object-oriented design. They described about the three following aspects of reuse: the interaction between some design processes, the mental processes involved in reuse and the mental representations constructed throughout the reuse activity. They also concentrated on software reuse benefits, types of reuse and reuse approaches. The two approaches mentioned by them for reuse of code are: identify and extract the reusable code from already developed code and develop the reusable code from scratch.

Danail Hristov, Werner Janjic [13] presented on the reusability of software components in an adhoc reuse scenario. It means the spontaneous decision of a developer to use a search engine, indexing or component repository. The results of this survey were separated for the sake of practicality i.e., the discovered metrics are divided into two categories: one for white-box (allowing to look into the code of the components) and one for black-box (where usually merely interface and documentation of a component are available) reusability. The reusability requirements for software components are explained and structured well in a reusability requirements model.

Neelamadhab Padhy, R. P. Singh1 [14] presented about new metrics which is the combination of one of the CK metrics suite and the one which calculates the reusable codes in the object oriented programming. Software measurement can be done in traditional or object oriented approach. The latest approach in the industry is an object oriented metrics through which the reusable components are created using inheritance. Some popular object oriented metrics which are

mentioned by them are developed in such a way to measure the reusable codes in the multi paradigm languages like (C++, Java Script, Ruby, Php, Perl, C# etc).

Edward A. Stohr [15] focused on the broad framework of software reusability research, and also suggests directions for future research. He presented about general, technical, and nontechnical issues of software reuse, and finally concluded that reuse needs to be viewed in the context of a total systems approach. He represented number of levels of abstraction, from abstract to concrete, in which both data and process entities are considered. He also presented about cost benefit and economic benefits models. Technical issues and organizational issues are clearly explained by him.

William W. Agresti [16] presented about a survey of 128 developers to explore their perceptions and experiences about using codes of other people. This also includes to what extent does the "not invented" here attitude exist? The survey was carried around a simple "4A" model, which is introduced in this article. The following are the four conditions must obtain to obtain benefits for reusing code: availability, accessibility, awareness and acceptability. This survey also includes the ways to take greater advantage of existing code and its related facts.

Manuel Sojer, Joachim Henkel [17] focused on open source software(OSS) which presents on how and why individuals and firms add to the commons of public OSS code i.e. on the "giving" side of this open innovation process. They also presented on how existing OSS code is reused and which serves as an input for further OSS development. It also mentions the interest of developers to handle difficult technical challenges as detrimental (tending to cause harm) to efficient reuse-based innovation.

Gui and Paul. D. Scott[18] presented about a set of new static metrics of coupling and cohesion which is developed to assess the reusability of Java components retrieved from the Internet by a software component search engine. These metrics differ from the majority of established metrics in the following three respects: they measure the degree to which entities are coupled or resemble each other, they quantitatively take account of indirect coupling and cohesion relationship and they also reflect the functional complexity of classes and methods. They focused mainly on issues in the development of such metrics for object-oriented systems.

Sarbjeet Singh, Sukhvinder singh [19] presented about the reusability concepts for Component based Systems and also discussed on several existing metrics for both white-box and black box components that can be used to measure reusability directly or indirectly and presents the special requirements on software in this domain. They also presented about glass box reusability. Components of reusability such as business component, distributed component, group component and software development component are discussed by them. Reusability models such as productivity model and maturity model are also discussed by them.

Adnan Khan, Khalid Khan[20] presented about a component based framework for software reusability which allows us to develop and integrate product components. This facilitate software reusability, high quality and simplification for testing. This component based framework uses kind of approaches which are based on object oriented design, architecture definition languages and software architecture. This speeds up software development by using already existing components, which in turn reduces cost and time.

Ravichandran[21] proposed a detailed systematic literature review on software reusability metrics for object oriented software program. Reusability is an only one best direction to increase developing productivity and maintainability of application. One must search for the better tested software component and reusable. Developed Application computer code by one coder may be shown helpful for others conjointly element. this may be proving that code specifics to application needs can be conjointly reused in develop comes connected with same needs. the main aim of this paper is to project some way for reusable module. A process that takes source code as a input that will helped to take the decision approximately, reusable artefacts ought to be reused or not.

A.N.Swamynathan and Dr.K.Nirmala[22] presented a comparative study on main and sub-code reusability. Most of the coding and reused coding of south Indian IT companies will be based on the object oriented programming surroundings (OOP)). While calling the class member function in objects of a particular class, interface and the dependency related problems are encountered. To beat these forms of issues, we have a tendency to propose a general purpose code reusable model that analyzes language structure through two attainable reusing environments. The common and ancient approach of this paper is that the main to sub-coding. It justifies the model based mostly approach for code reusability under OOPs for these two approaches.

Stephan Fischer [23] proposed the reusability of interactive resources in web based educational systems. Reusability can be triggered by a mixture of these reusable transmission elements and therefore the applicable use of information to regulate the elements similarly as their combination. in this article, we tend to discuss the reusability and flexibility aspects of multimedia content in web-based learning systems. In distinction to existing approaches, we tend to extend the component-based design to create multimedia visualization units with the utilization of information for reusability and customizability.

Richard T. Vaughan [24] proposed the device abstractions for portable, reusable robot code. This paper describes the appliance of three acknowledge abstractions, the character device model, the interface/driver model, and also the client/server model to the present purpose. One product of this project is that the Player Abstract Device Interface (PADI) specification which defines a group of interfaces that capture the practicality of logically similar sensors and actuators. This ficapteri is that the central abstraction that allows Player-based controllers to run unchanged on a range of real and simulated devices. we have a tendency to propose that PADI may well be a place to begin for development of a typical platform for mechanism interfacing, freelance of Player, to change code movability and re-use, whereas still providing access to the distinctive capabilities of individual devices.

AshishAgrawal [25] presented an approach to analysis software reusability. The degree that could be a software Reusability module or early product work will be used inadditional than one system computing or software

program. It is a belief that software reusability provides the key to tremendous advantages and saving in software development product. The U.S. Department of Defense solely can be save $300 million annually by increasing its level of reuse by as very little as one percent. Measurement might facilitate U.S.A. not solely to be told the way to build reusable parts however additionally to spot reusable parts among the wealth of existing programs. Existing programs contain the information and knowledge gained from operating within the specific application domain and meeting the organization's software needs.

William B. Frakes and Thomas P. Pol[26] proposed the methods for reusable software components. An empirical study of strategies for representing reusable computer components are described. Thirty-five subjects searched for reusable parts in a very informative of operating system tools exploitation four totally different illustration methods: attribute-value, enumerated, faceted, and keyword. The study used Proteus, an employ library system that supports multiple illustration strategies. looking effectiveness was measured with recall, precision, and overlap. Search time for the four strategies was additionally compared. Subjects rated the strategies in terms of preference and helpfulness in understanding parts. Some principles for constructing employ libraries, supported the results of this study are mentioned.

Deng-jyi Chen [27] presented the integration of reusable software components. Software reuse is an efficient suggests that of improving the software productivity and software quality. Reusable computer elements of code (RSCs) are the fundamental building elements for critical software programs created mistreatment the software utilize approach. The object-oriented approach is employed to style and implement our RSCs. multimedia system software plays a vital role within the software business. In contrast to ancient software, multimedia system software provides users with visual and audio effects through their interfaces and may have a lot of accurately model the real world.

James Solderitsch, Elizabeth A. Bradley [28] proposed an approach i.e. reusability library framework(RLF). Reducing unnecessary and redundant system development through reuse is a key element in the Department of Defense's Software Technology Strategy to reduce the annual DoD software cost. The STARS1 program is implementing this strategy with an approach that combines reuse with other elements including the identification of software processes along with a focus to specific application domains. The Reusability Library Framework(RLF), being developed and employed by Electronic systems/Valley Forge Laboratories in cooperation with Tactical Systems, addresses these needs. This paper will describe the current status of the RLF and give examples of how it has been applied to date to several paramax application areas.

Li Jingyue [29] presented a survey on software reuse. Software reuse means that reusing the inputs, the processes, and therefore the outputs of previous software development efforts.

Although there are several studies during this space, there still some fields to be explored. Four fields, that is, domain engineering, business line, and component based software engineering and COTS square measure mentioned during this paper. Challenges in every field are generalized and summarized during this paper so as to own a full image of the state of the art of software reuse.

Catherine Blake Jaktman [30] presented about the software reuse. Software reuse has come to mean anything from the ad-hoc process of copying and modifying code that basically achieves the required task to the more systematic reuse of various forms of software development experience. Organizations often attempt to implement a software reuse program without an adequate understanding of their capability for doing so effectively. This paper describes an initial reuse assessment that can assist organizations in planning their reuse strategy based on their current infrastructure, reuse goals, and availability and quality of existing reusable assets.

Judith Barnard [31] proposed that, the managers of software companies came to know that the reuse of code can bring financial rewards to their companies. So that many companies are going to start their own reuse libraries. The factors that are related to the reusability of code which are identified by metrics are coupling between objects, method correctness and method interface complexity. The reusability of code metrics which can be used to provide a value of reusability for a class from any programming language and it can also be used to guide the programmer for writing more reusable code.

William Frakes and Carol Terry [32] viewed that software reuse is the program which is developed by the organizations to improve its productivity and quality. We can identify the most reuse strategies by reuse metrics and reuse models. A metric is a quantitative indicator for an attribute. A model specifies relationships between the metrics. The metrics will provide good management information for a library system. Metrics can also be used in various ways in-order to demonstrate the value of the library to management as well as to provide information for quality improvement.

M.Ramesh and H.Raghav Rao [33] defines the reuse and the categories in it in this model. The reusable code is not very complicated.A Reuse Support System (RSS) should be able to find and retrieve the components according to the givencriteria, evaluate the retrieved components according to the given constraints, and rank the each evaluatedcomponent in some order. A classification scheme is needed in order to find and retrieve the components.

William B. Frakes [34] declares that in this model we came to know the factors that affect the reuse, the reuse measurement, and the techniques for tailoring a reuse program to a given organization via a failure modes model. The software reuse improvement strategies can be created and fine-tuned for an organization. This improves the productivity, reliability, better bid estimation, better early estimates, and faster time to market.

W B Frakes and P B Gandel[35] proposed an approach that relates the reusable methods to one another and also to the software life-cycle model. A software representation is a mapping of predicates and the terms that describes the individual objects and the relationships among objects from the represented to the representing world. The main aim of

the reusable library developers is to provide various ways for software users to search for the software components that satisfy the user needs.

J E Gaffney, Jr, and T aDurek [36] presented that due to the software and code reuse there is a reduction in development and overall software life-cycle costs. It enables us to build more software and reduce application backlogs. It improves the enhance quality. Therefore, it results in the saving of money and enhancing the productivity. The quantitative and qualitative models are employed to estimate its impact on time, schedule, and required to develop a software based on the extent of reusing involved in the construction.

Tuomas Ihme [37] presented that a software reuse system can be characterised by input specifications, output models, constraints from requirements and technologies to support reuse. We can make it more flexible by getting the feedback from the actual user. The important activities in the software reusability process in each specification and every design stage are creating, focus, locate, adjust, and configure.

Patrick A V Hall [38] presented that software reusability fulfills the user needs. Software reuse also reduces the work of repeatedly doing the tasks in detail the same constructs to achieve the same end. To make reusable of software, it must be able to specify the software with sufficient precision and completeness for a potential user to be possible to make his selection. The reusing of software includes the algorithms and methods, languages and systems etc.

Jaime Nino [39] presented that the development method for generating the reusable software is incremental software development. A concept of modular software is needed for the support of incremental software development through the refinement of the existing software modules. Object Oriented Programming Languages provide us with several design and implementation alternatives of this flexible modular structure.

Capers Jones T [40] presented that the first concept of reusable modules was a single function that could line by line: functions such as square root extraction, date conversions, and statistical routines have been utilized since the early days of computing. The attempts when creating standard reusable modules have faced 3 major obstacles: i.) the problem of data reusability should be solved, ii.) an architecture for reusability should be created, and iii.) reusable designs should be created.

III.CONCLUSION

On analyzing these various techniques, the most important technique is Component Based Software Engineering (CBSE), it proposes to mine reusable components from a set of similar object-oriented software, which were developed in the same domain, ideally by the same developers. Even though we have analyzed and compared various techniques to reuse the code, the new set of the code that needs to be programmed is designed in such a way that it should possess certain level of code reusability for the future client, which is unpredictable. An unpractical design in this case will go for complete loss of production and may not meet the reusability factor for new projects. Hence, its future direction of study will focus on estimating the level of code reusability for complex software projects, anticipating that proposed design concept will highly encourage and motivate the stakeholder to consider it as most cost-effective tool till date.

IV. ACKNOWLEDGEMENT

The authors wish to acknowledge prof. Lavanya.K for her guidance in research work. We would also like to express our gratitude for her valuable time and assistance given in helping us to write this review paper.

V.REFERENCES

1.Er. Bhupinder Singh, Er. Satinder Pal Ahuja "Detection of reusable components in object oriented programming using quality metrics" International Journal of Advanced Research in Computer Science and Software Engineering,Volume 3, Issue 1, January 2013 ISSN: 2277 128X

2. Sandeep Dalal,Dhreej Chhillar "An effort to improve cohesion metrics using inheritance" International Journal of Computational Engineering Research,Vol 03,Issue 6,June 2013

3. Priti Srinivas Sajja "Automatic generation of agents using reusable soft computing code libraries to develop multi agent system for healthcare" I.J. Information Technology and Computer Science,05, 48-54,Published Online April 2015 in MECS (http://www.mecs-press.org/),DOI: 10.5815/ijitcs.2015.05.07

4. Kuntal Barua and Dr. M. K Rawat "a minimization of software cost and effort estimation using code reusability concept by analogy estimation technique" Engineering Universe for Scientific Research and Management,(International Journal),Impact Factor: 3.7,Vol. 6 Issue 4,April 2014

5. Abdelhak-Djamel Seriai "mining reusable software components from object-oriented source code of a set of similar software" 1This work has been funded by grant ANR 2010 BLAN 021902.

6.Hlomani Hlomani, "towards an algorithms ontology cluster: for modular code reuse and polyglot programming" ACSIJ Advances in Computer Science: An International Journal, Vol. 5, Issue 2, No.20, ISSN: 2322-5157, March 2016.

7.Nandakumar A.N, "constructing relationship between software metrics and code reusability in object oriented designAPTIKOM Journal on Computer Science and Information Technologies,Vol. 1, No. 2, pp. 59~72,ISSN: 2528-2417,DOI: 10.11591/APTIKOM.J.CSIT.16,2016,

8. Suresh Satapathy and R. P. Singh, "utility of an object oriented reusability metrics and estimation complexity" Indian Journal of Science and Technology, Vol 10(3), DOI: 10.17485/ijst/2017/v10i3/107289, ISSN (Print): 0974-6846, ISSN (Online): 0974-5645, January 2017.

9. Priti Srinivas Sajja, "Computer aided development of fuzzy, neural and neuro-fuzzy systems" International Journal of Engineering and Applied Computer Science (IJEACS) ISBN: 978-0-9957075-2-8,Volume: 02, Issue: 01, January 2017.

10.Dr. Vivek Chaplot, "study & analysis of object oriented programming principles",International Educational and Research Journal,E-ISSN No:2454-9916,Volume:1,Issue:4,Nov 2015

11. V. Subedha, Dr. S. Sridhar "A Systematic Review of Reusability Assessment Model and Related Approach for Reusable Component Mining" Journal of Computer Applications (JCA)ISSN: 0974-1925, Volume V, Issue 2, 2012.

12. Neha Budhija and Satinder Pal Ahuja "Review of software reusability"International Conference on Computer Science and Information Technology (ICCSIT'2011) Pattaya, *Volume VI– No.10,* Dec. 2011

13. Danail Hristov, Oliver Hummel, Mahmudul Huq, Werner Janjic"Structuring Software Reusability Metrics for

Component-Based Software Development" Software Engineering Group, University of Mannheim, Mannheim, Germany, *Vol 10(3), DOI: 10.17485/ijst/2010, Nov. 2011*

14. Neelamadhab Padhy1, Suresh Satapathy2 and R. P. Singh1 "Utility of an Object Oriented Reusability Metrics and Estimation Complexity" Indian Journal of Science and Technology, Vol 10(3), DOI: 10.17485/ijst/2017/v10i3/107289, January 2017
15. Edward A. Stohr "Software Reuse: Issues and Research Directions" Indian Journal of Science and Technology,*Volume 9– No.9, October*, 2010
16. William W. Agresti "Software Reuse: Developers' Experiences and Perceptions" journal of software engineering and applications, 2011, 1, 48-58**,** doi:10.4236/jsea.2011.41006 Published Online January 2010 (http://www.scirp.org/journal/jsea)
17. Manuel Sojer1 and Joachim Henkel "Code Reuse in Open Source Software Development: Quantiative Evidences, drivers and Impidents" Centre for Economic Policy Research (CEPR), London, *Volume 10– No.5, November 2010*
18. Gui and Paul. D. Scott "Measuring software reusability component by coupling and cohesion metrics" JOURNAL OF COMPUTERS, VOL. 4, NO. 9, SEPTEMBER 2009
19. Sarbjeet Singh, Manjit Thapa, Sukhvinder singh and Gurpreet Singh "Software Engineering: Survey of reusability based on software component" International Journal of Computer Applications (0975 – 8887) *Volume 8– No.12, October 2010*
20. Adnan Khan, Khalid Khan, Muhammad Amir and M. N. A. Khan "Component based framework for software reusability" International Journal of Software Engineering and Its Applications, Vol. 8, No. 10 (2014), pp. 13-24 http://dx.doi.org/10.14257/ijseia.2014.8.10.02, Oct 2009
21. Anthes, Gary I "Software Reuse Plans BringPaybacks," Compute world, Vol. 27, KO. 49, pp.7376. 22.G.Chin, "Agile Project Management: How to Succeed in the Face of Changing Project Requirements" American Management Association (AMACOM), New York, USA, 2004.

23.Roschelle, J., Digiano, J., Koutlis, M., Repenning, A., Phillips, J., Jackiw, N., and Suthers, D. 1999. developing educational software components. IEEE Computer 32, 9 (Sept.), 50–58, Dec. 2002

24.Brian P. Gerkey, Richard T. Vaughan, and Andrew Howard. Player User Manual 1.3. Player/Stage Project, http://playerstage.sourceforge.net, December 2002.

25. Ashish Agarwal "Software Reusability: Applications and Experiences", IEEE computer32, Vol II, Addison Wesley, 1999.

26.R. Prieto-Diaz, "Implementing faceted classification for software reuse," Communication ACM, vol. 34, pp. 88-97, Oct 1999.

27.R. E. Johnson, "How frameworks compare to other object-oriented reuse techniques: Frameworks = Components + Patterns," Communications of the ACM, Vol. 40, No. 10, 1997, pp. 39-42.

28. Ronald j.brachman and j.schmolze "An overview of the kl-one knowledge representation system" cognitive science,9(2),171-216, Nov 2001.
29. Colin Atkinson, Joachim Bayer, Christian Bunse, Erik Kamsties, Oliver Laitenberger, Roland Laqua, Dirk Muthig, Barbara Paech, Jürgen Wüst, Jörg Zettel, "Component-based Product Line Engineering with UML", Vol. 20, No. 10 Addison-Wesley, 2002.
30. Frakes, W.B. and Isoda, S. "Success Factors of Systematic Reuse," IEEE Software,pp. 15-18, September 2004
31. Judith Barnard "New reusability metric for object-oriented software", *Software Quality Journal* 7, (1998) 35–50

32.William Frakes and Carol Terry "Software Reuse: Metrics and models"1996, ACM Computing Surveys, Vol. 28, No. 2, June 1996

33. M.Ramesh and H.Raghav Rao "Software Reuse: Issues and an example" Decision Support Systems 12 (1994) 57-77 57,North-Holland
34. William B. Frakes "Software Reuse: An Empirical Approach" Annual *Review of Automatic Programming* Vol. 16. pp. 41-44, 1992
35. W B Frakes and P B Gandel "Representing Reusable Software" Vol 32 no 10 December 1990
36. J E Gaffney, Jr, and T A Durek "Software Reuse – Key to Enhanced Productivity: Some Quantitative Models" journal of information and science technology,1989
37. Tuomas Ihme "A Reuse Base for Real-Time Software Specifications"North- Holland Micro processing and Microprogramming 27 (1989) 639—646
38. Patrick A V Hall "Software Components and Reuse – getting more out of your code" vol 29 no I January/February1987
39. Jaime Nino "Object Oriented Models for Software Reusability"IEEE Conference Publications,vol.2, 395-399.
40. Capers Jones T "Reusability in programming: A survey of the State of the Art" IEEE Transactions on software engineering, vol. se-10, no. 5, September 1984.

