Journal of Theoretical and Applied Information Technology
10th June 2015. Vol.76. No.1
© 2005 - 2015 JATIT & LLS. All rights reserved·

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ALL ABOUT SOFTWARE REUSABILITY: A SYSTEMATIC LITERATURE REVIEW

[1]SIHAM YOUNOUSSI, [2]OUNSA ROUDIES

[1,2]Mohammed-V Agdal Univ, Ecole Mohammadia d'Ingénieurs (EMI), Siweb Research Team.

E-mail: [1]siham.younoussi@gmail.com, [2]roudies@emi.ac.ma

ABSTRACT

Software reusability is an attribute in which software or its module is reused with very little or no modification. For any organization, improving the business performance means performing their software development. Software reusability offers great potential of significant gains for an organization, by reducing cost and effort, and accelerating the Time to Market of software products. This paper presents a literature review of various software reusability concepts. It presents some definitions and benefits of software reusability, approaches to be adopted to perform reusability, reusability levels in software life cycle, to reusability, maturity models and attributes affecting potentiality of software to be reused.

Keywords: *Software Reusability Approaches, Software Reusability Benefits, Software Reusability Levels, Software Maturity Models for Reuse, Software Reusability Attributes*

1. INTRODUCTION

Reuse is one way for improving software development performance. That is why many organizations try to invest in software reusability, by identifying best reuse strategies, methods and component for maximum productivity.

Software reuse is creating new software systems, while reusability is the degree to which a given software component can be reused [1]. According to [2], reusability is a property of a software component that indicates its capability of reuse.

Software reuse is the process of building software system from existing software rather than building them from scratch [3]-[4]. Software reusability is an attribute that refers to the expected reuse potential of a software component [5]. This means that, if a component's reusability is low, then its potential for reuse becomes low as well.

According to [6], software reusability relates to using formerly written software in the form of specification, design and code. This practice is widely observed in the process of development for most projects as it brings about several advantages.

Although reusable components like design patterns, frameworks, component based software development (CBSD), are already popular in organizations, software reuse has rapidly evolved in the last decade, and new reusability approaches are emerging. So mastering reuse is necessary to simplify and to foster reusability in software development.

In this context, this paper focuses on trends in software reuse practices and aims to outline how reusability could improve the long-term organization.

2. RESEARCH METHOD

We conducted a systematic literature review to understand and identify approaches, benefits, levels, barriers, maturity models and attributes, of software reusability.

A systematic literature review (SLR) is a mean of identifying, evaluating and interpreting all available research relevant to a particular research question or topic area [7]. The first step are eliciting the research questions and mastering the quality of collected papers.

2.1 Research Questions

We point out six research questions (RQ).

RQ1: What are the different approaches of the software reuse?

This question aims at identifying the current approaches of software reusability. We analyzed the main definitions, goals and advantages of these approaches.

RQ2: Are there any benefits of software reusability?

It was important for us to know what the benefits of software reuse are, and clarify why Organizations are looking for ways to develop a software reuse schedule.

RQ3: What are the different levels of software reusability in software life cycle?

We need to know what the different reusability levels are in software life cycle, and how it is applied in these levels.

RQ4: Are there any barriers of software reusability?

The goal is to identify barriers of software reusability that must be overcome to successful reuse.

RQ5: What are the maturity model for reuse?

We were looking for researches and case studies that proposed maturity model for reuse.

RQ6: Is there any attributes that affect the software reusability?

The purpose of this question is to elicit attributes affecting the software reuse and relate to the potentiality of software to be reused.

2.2 Research Process

We started our research process of identifying primary studies by searching on the electronic databases for researches that cover almost all major journals and conference proceedings. The repositories used were ACM Digital Library, IEEE Xplore, Science Direct, Springer, and Scopus.

Based on our research goal, the following major search keywords were used to formulate the search query: Software reusability, approaches, benefits, levels, maturity models and attributes. Alternative words and synonyms were also used for such keywords. Then, it was created an initial pilot search string by joining major keywords with Boolean AND operators, and the alternative words and synonyms with Boolean OR operators.

2.3 Study Selection Criteria

Study selection was performed in the first step by analyzing the title and abstract of articles found in search process and select appropriate and relevant studies. In the second step, we focused on analyzing the introduction and conclusion. The inclusion and exclusion criteria were analyzed for each step in each primary study.

The inclusion and exclusion criteria used in our study, are the following:

- ***Inclusion criteria:***
  - Papers discussed about software reusability approaches and benefits.
  - Papers discussed about maturity model of reuse.
  - Papers published from 2004 to 2015.
- ***Exclusion criteria:***
  - Papers out of our research scope.
  - Short papers of one or two pages.
  - Repeated papers.

2.4 Quality Assessment

The quality assessments are based on a checklist of factors/questions that needs to be evaluated in each study. For assessing studies, we defined the following questions:

QA1: Does study mention the software reusability approaches?

QA2: Does the study presented any benefits of software reusability?

QA3: Is the study list the different levels of software reusability?

QA4: Does study report any barriers of software reusability?

QA5: Does study propose any maturity model for reuse?

QA6: Does study propose any attributes that affect the reuse?

We scored questions as bellow:

QA1: Y (Yes) study proposed some software reusability approaches. P (Partially) study mentioned one or more approaches, but did not describe it. N (No) study did not propose any approaches.

QA2: Y, study mentioned more than one benefits of software reusability clearly. P, benefits are implicit. N, study does not mention any benefit.

QA3: Y, study defined some levels of software reusability. P, reusability levels are implicit. N, study did not present any levels.

QA4: Y, study mentioned some barriers of reusability explicitly. P, reusability barriers are implicitly reported. N, study did not report any barriers.

QA5: Y, study proposed some maturity models for reuse. P, study mentioned one or more maturity model, but did not describe it. N, study did not propose any maturity model.

QA6: Y, study mentioned attributes which affect the reuse explicitly. P, attributes are implicit. N, study did not mention any attribute.

2.5 Data Collection

These data were extracted from each paper:

- Title and year of publication
- Author(s) information
- Research issues
- Main topic
- The full source and references

All articles were reviewed and data was extracted and checked. This idea was chosen for better consistency in reviewing all papers and improving quality of review.

RESULTS

This section summarizes the results of our study.

3.1 Search Results

As a result of performing study selection in the first step, we get 252 papers. Title and abstract of these papers were analyzed by applying the including and excluding criteria, and then the number of papers became 112.

In the second step, introduction and conclusion of the 112 papers were evaluated using including and excluding criteria. The final number papers selected in this review was 24 papers as shown in fig.1, and final selected studies are listed in table 1.

3.2 Quality Evaluation of Studies

We assessed the studies for quality using the criteria explained in section 2.4, and the scores for each of them are shown in table 2.

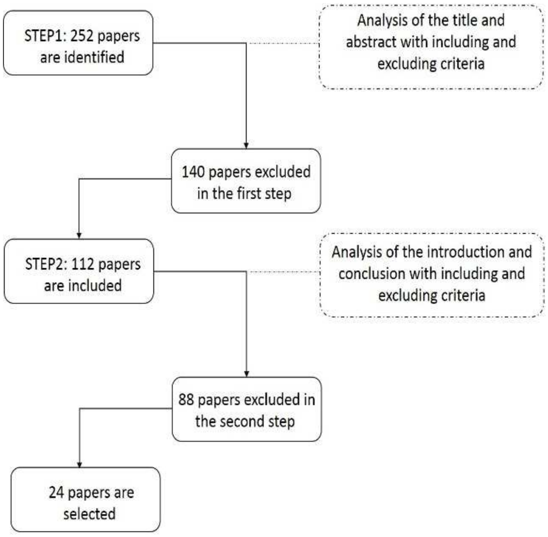
STEP1: 252 papers are identified
Analysis of the title and abstract with including and excluding criteria
140 papers excluded in the first step
STEP2: 112 papers are included
Analysis of the introduction and conclusion with including and excluding criteria
88 papers excluded in the second step
24 papers are selected

Fig. 1. Selection research process

Table 1: Selected Studies For Review

ID	Title	Authot(s)	Main topic	Year
S1	Rise project: Towards a robust framework for software reuse	E. S. Almeida, A. Alvaro, D. Lucr´edio, V. C. Garcia, and S. R. L Meira,	Overview of the RiSE Maturity Model, which has been developed within the RiSE project	2004
S2	Reusability Metrics for Software Components	O. Paul ROTARU, Marian DOBRE	Study of the adaptability and compose-ability of software components, with proposing metrics and a mathematical model for the above-mentioned characteristics of software components.	2005

S3	Software Reuse Research: Status and Future	B. William. Frakes and Kyo Kang	Summarized software reuse research, discusses major research contributions and unsolved problems	2005
S4	A Forward-Looking Software Reuse Strategy	J. Finnigan, J.Blanchette	Described a software reuse strategy and illustrate that strategy using the command-building software as an example.	2007
S5	Towards a Maturity Model for a Reuse Incremental Adoption	V. Garcia, D. Lucrédio, A. Alvaro	Reuse Maturity Model proposal, describing consistence features for the incremental reuse adoption	2007
S6	Knowledge reuse for software reuse	F. McCarey, M.O. Cinneide and N.Kushmerick	Component-based reuse can be supported through knowledge collaboration	2008
S7	Reducing efforts on software project management using software package reusability.	R. Kamalraj, B.G Geetha, G. Singaravel	Focused on the consecutive tasks like 'Domain Analysis', 'Package Analysis;' and 'System Analysis' for reusability to minimize the required technical efforts in development area	2009
S8	Reusability assessment for software components	A.Sharma, P.S. Grover and R. Kumar	Artificial neural based approach is been proposed to access the reusability of software component	2009
S9	A Value Analysis Model for Measuring Software Reuse	M. Dinsoreanu, I. Ignat	Presented an integrated measurement model that allows practitioners to apply familiar project management techniques for measuring software reuse and to include software reuse metrics in the analysis of project performance indicators.	2009
S10	A survey on software reusability	P.S. Sandhu, P. Kakkar, S. Sharma	Presented the reusability concepts for Component based Systems and explores several existing metrics	2010
S11	Overview analysis of reusability metrics in software development for risk reduction	G. Singaravel, V. Palanisamy, A. Krishnan	Provided a reusability metrics in software development for risk reduction, because risk is directly proportional to the complexity of a system and risk is inversely proportional to the number of reusable components used in a project.	2010
S12	A New Capability Maturity Model For Reuse Based Software Development process	K.S Jasmine, R. Vasantha	Approach for making CMMI investment decisions by proposing a new process based capability maturity model for reuse.	2010
S13	Software reusability assessment using soft computing techniques	Y. Singh, P.K. Bhatia and O. Sangwan	Proposed a model for accessing software reusability by different soft computing techniques	2011
S14	Software Reuse in Agile Development Organizations - A Conceptual Management Tool	W. Spoelstra, M. Iacob, M. Sinderen	A conceptual management tool for supporting software reuse is proposed.	2011
S15	Designing code level reusable software components	B.Jalender, A. Govardhan,R. Emchand	Described how to build the code level reusable components and how to design code level components	2012
S16	Reusability of Software Components using J48 Decision Tree	K. Kaur, N. Mohan and P. S. Sandhu	Proposed a reusability of Software Components using J48 Decision Tree	2012
S17	Component-Based Development: A Unified Model Of Reusability Metrics	B. Koteska, G. Velinov	Proposed new metrics for component reusability	2013

S18	Minimal information for reusable scientific software	C. Hong	Looks at the concept of software reusability from the perspective of the software engineer and the researcher.	2014
S19	Reusability in Component Based Software Development - A Review	S. Thakral, S. Sagar and Vinay	A literature review of various software reusability concepts is presented	2014
S20	Software Reuse in Practice	R. Keswani, S. Joshi, A. Jatain	Summarized software reuse research and discussed major research contributions.	2014
S21	Impact of Quality Attributes on Software Reusability and Metrics to assess these Attributes	C. Monga, A. Jatain, D. Gaur	Studied various attributes or factors that affect the reusability of software. The most common factors are identified and their impact is analyzed.	2014
S22	Taxonomy, Definition, Approaches, Benefits, Reusability Levels, Factors and adaptation of Software Reusability: A Review of the Research Literature	Y. Y. Ibraheem, A. M. Abualkishik and M. Z. Mohd Yussof,	Provided a systematic review of the concept of reusability, identifying the definition, Approaches, Benefits, Reusability Levels, Factors and adaptation of Software Reusability	2014
S23	Feature Prioritization for Analyzing and Enhancing Software Reusability	Md. Iftekharul A. Efat, Md. S. Siddik, M. Shoyaib, S. M. Khaled	An analysis of the various attributes from the organization, development and complexity perspective, an optimized group of properties are proposed	2014
S24	A Framework for Assessing the Software Reusability using Fuzzy Logic Approach for Aspect Oriented Software	P. K. Singh, O. P Sangwan, A. P. Singh, A. Pratap	Explored the various metric that affects the reusability of aspect oriented software and Estimate it using fuzzy logic approach.	2015

Table 2: Quality Evaluation of the Study

Source	QA 1	QA 2	QA 3	QA 4	QA 5	QA 6
S1	Y	Y	P	Y	P	P
S2	Y	N	Y	N	N	Y
S3	P	Y	N	N	N	Y
S4	Y	N	N	N	N	P
S5	Y	Y	Y	N	Y	P
S6	Y	Y	N	N	N	N
S7	Y	P	P	N	N	Y
S8	Y	P	P	N	N	Y
S9	N	Y	N	N	P	P
S10	Y	Y	P	Y	Y	Y
S11	Y	Y	Y	N	N	Y
S12	N	Y	Y	N	Y	N
S13	N	N	Y	N	P	Y
S14	Y	Y	P	Y	Y	Y
S15	Y	Y	Y	N	P	P
S16	Y	Y	Y	Y	N	Y
S17	N	P	Y	P	Y	P
S18	N	Y	N	N	Y	P
S19	P	Y	Y	Y	N	Y
S20	N	Y	P	Y	N	N
S21	Y	P	N	N	N	Y
S22	Y	Y	Y	P	N	Y
S23	Y	Y	P	N	N	Y
S24	Y	Y	N	N	N	Y

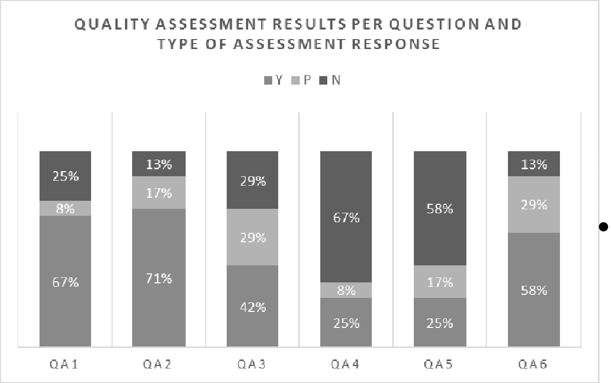

Fig. 2. Quality assessment results per question and type of assessment response

Fig. 2 shows the coverage of every quality assessment (QA) in the included studies. It shows that QA1, QA2, QA3 and QA6 were covered in a rate higher than 80% by Yes and partially answers. That means that 80% of studies covers approaches, benefits, levels and attributes of software reusability. On the contrary, QA4 and QA5 were covered in a rate higher than 50% by No. Which means that few works examined barriers of reusability (QA4), which can motivate organizations to adopt software reusability approaches. Moreover, the studies about maturity models to software reusability are limited, which highlight the need to explore this domain in order to help organizations auditing his maturity reuse levels.

3. DISCUSSION

This section discusses the answers to the six research questions.

4.1 Does Study Mention the Software Reusability Approaches?

75% of studies presents different software reusability approaches used by developers. There are eleven software reusability approaches [3]-[5] which are the most uses: Application Frameworks, Application product lines, Aspect-oriented software development, Component-based development, Configurable vertical applications, COTS (Commercial-Off-The-Shelf) integration, Design Patterns, Legacy system wrapping, Program generators, Program libraries and Service-oriented systems.

- **Application Frameworks**: are collections of concrete and abstract classes that can be adapted and extended to create application systems. Application frameworks are reusable software products that deliver reusable design and common implementation to applications of a specific domain.
  The main advantages for Application Frameworks are reducing the development cost, enhancing the quality, promoting the software reusability benefits and reusable design.
- **Application product lines**: A product line is defined as a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way [28].
  Software product line has proven to support systematic reuse across the set of similar products that software companies offer.
  The main advantages for adopting Application product lines have discussed by several authors [28]-[3]. It usually include developing products more efficiently, get them faster to the market faster to stay competitive and produce with higher quality.
- **Aspect-oriented software development:** Aspect-oriented software development (AOSD) is a new approach to software development that addresses limitations inherent in other approaches, including object-oriented programming. AOSD aims to address crosscutting concerns by providing means for systematic identification, separation, representation and composition. Crosscutting concerns are encapsulated in separate modules, known as aspects, so that localization can be promoted. This results in better support for modularization hence reducing development, maintenance and evolution costs [27].
  The main advantages for Aspect-oriented software development are increasing the software quality, enhancing the development mechanisms, automating the mapping from problem to solution and increasing modularity.
- **Component-based development:** The main idea of the component-based approach is building systems from already existing components. This assumption has several benefits: enhance efficiency, enhance the ability to reuse components, managing growing complexity, reducing the time and effort needed to develop software, decreasing production costs through software reuse, enhancing the quality of system, reducing maintenance costs, increasing development productivity[6].

The main advantages for Component-based development are reducing the development time and cost, improving the software quality and maintainability.

- **Configurable vertical applications**: Configurable vertical application is a generic system that is designed so that it can be configured to the needs of specific system customers [4]. An example of a vertical application is software that helps doctors manage patient records, patient and insurance billing. Vertical application is considered as a system that is concentrated on a narrow set of simulation.
  The main advantages for Configurable vertical application are facilitating the configuration, boosting the development, and anticipating the future needs of users.
- **COTS (Commercial-Off-The-Shelf) integration:** COTS products are designed to be implemented easily into existing systems without the need for customization, products that are ready-made and available for sale to the public. The easiest method to develop systems quickly with lesser cost than the traditional development is using development by integration of pre-fabricated COTS components, so COTS integration is considered as a type of application system reuse.
  The main advantages for COTS integration are reducing the development effort and cost, improving the software quality, and increasing the maintainability of the system.
- **Design Patterns:** A design pattern is a solution for a recurring problem in software engineering. A design pattern is a template for how to solve a problem that can be used in many different situations [6].
  The main advantages of Design patterns are increasing the flexibility for potential changes, increasing productivity and software reusability benefits, and reducing design problems.
- **Legacy system wrapping:** By wrapping a set of defining interfaces by legacy systems provides access to interfaces. Rewriting a legacy system from scratch can create an information system with equivalent functionality, and based on modern software techniques and hardware [3].
  The main advantages for Legacy system wrapping are allowing access to interfaces, reduce cost, and help to make the wrapping process automatic for reducing user intervention.
- **Program generators**: A Program Generator is a program that enables an individual to create program of their own easily with less effort and programming knowledge. With a program generator a user may only be required to specify the steps or rules required for his or her program and not need to write any code or very little code A generator system embeds knowledge of a particular type of application and can generate systems or system fragments in that domain. Program Generators Involves the reuse of standard patterns and algorithms [3].
  The main advantages for Program Generators are reducing the development efforts and cost, improving the development quality, and accelerating the development.
- **Program libraries**: Function and class libraries implementing commonly used abstractions are available for reuse. Libraries contain data and code that provides necessary services to independent programs. This idea encourages the exchanging and sharing of data and code [3].
  The main advantages for Program libraries are enhancing the quality, reducing of system errors, boosting the reuse, and boosting the sharing of code and data.
- **Service-oriented systems**: is a set of methodologies and principles for developing and designing software in the form of component. These components are developed by linking shared services that may be externally provided. An enterprise system often has applications and a stack of infrastructure including databases, operating systems, and networks [3].
  The main advantages for Service-oriented systems are offering a more flexible method for software development, offering a better reuse, and allowing software systems to be dynamic.

4.2 Are there any Benefits of Software Reusability?

88% of the article agree that there are many benefits of software reusability. By reviewing them, it seems that the authors agreed that the major benefits are the following:

- **Increase productivity:** Software reusability improves productivity because the existing software products are using, and very fewer ones are creating from scratch.

According to Singh [4], the concept of reusing for an available software components consider as a key feature in increasing productivity.

- **Minimize cost**: The cost of developing software from scratch can be saved by identifying and extracting the reusable components from already developed and existing systems or legacy systems [4].
- **Improve quality**: A good software reuse assists the increasing of reliability and quality [3].
  According to [24], improve software quality using any software over time reflects many bugs which were not discernible when it was created. Therefore, a software product that has been reused many times will contain much less bugs and defects than freshly created software.
- **Increase dependability:** Increasing dependability will reduce the time of the software development since it minimize the development failures [16].
  Reused software, that has been tried and tested in working systems, should be more dependable than new software [6]
- **Accelerate development**: Reusing software can speed up system production because both development and validation time should be reduced [16].
  According to [17] Reusing software to build a new software product can reach the market by on time for satisfying the customer needs. Generalized software components can minimize the time of product construction and delivery of software.
- **Reduce process risk**: If software exists, there is less uncertainty in the costs of reusing that software than in the costs of development. This is an important factor for project management as it reduces the margin of error in project cost estimation. This is particularly true when relatively large software components such as sub-systems are reused [16].
  According to [17] Risk in creating new software is reduced when available reusable components already encompass the desired functionality and have standard interfaces to facilitate integration.

4.3 Is the Study List the Different Levels of Software Reusability?

71% of articles proposed explicitly or implicitly different reusability level in software life cycle. Reuse is divided into following levels:

- **Specification reuse:** Understanding what to build is one of the most tedious aspects of software development, because sometimes customers do not really know what they want, so capitalizing on previously used abstract artifacts like requirement specification document may open the mind of software customers to more functionalities that could have been overlooked.
  The reuse of specification is considered as a higher level of reuse [6]
- **Design reuse:** The design process in most engineering disciplines is based on reuse of existing systems or components.
  Software reusability more specifically refers to design features of a software element (or collection of software elements) that enhance its suitability for reuse [16]. This type of reuse is required when a system needs to be reported in an entirely different software or hardware environment [6].
- **Code reuse**: In computer science and software engineering, reusability, is the likelihood a segment of source code that can be used again to add new functionalities with slight or no modification [16].
  The reusability of a piece of code does not mean that we should be able to copy-paste the same code in many places within the application. In fact, it exactly means the opposite thing. A piece of reusable code means that the same code can be reused in different places without re-writing it [17].
  The reusable code can be object code, data objects, source code, and standard subroutines [6].
- **Application system reuse**: An increasing number of organizations are using software not just as all-inclusive applications, as in the past, but also as component parts of larger applications. In this new role, acquired software must integrate with other software functionality [16].
  Application system reuse is considered a special case of software reuse, where the whole system is reused by implementing in through a range of different operating systems and computers [6].
- **Test reuse**: Reusable components are usually accompanied by high quality documentation and by previously developed tests plans and cases. Whenever a new system is created by simple selection and altering of such components, then, their documentation and

tests will have to be much easier to develop as well.

4.4 Does study report any barriers of software reusability?

We cannot ignore the significant benefits provided by systematic software reuse, but its implementation is not simple, because many factors make it infeasible, particularly in companies with a large installed base of legacy software and developers [S20].

Only 33% of studies depicts barriers of software reusability. Various barriers are identified that must be overcome such as the following:

- **Organizational barriers:** to reuse software one needs a deep understanding of application developer needs and business requirements only then one can develop and deploy old software for reuse [24].
- **Administrative barriers**: Owing to the large size of industry, it becomes very difficult to reuse software or part of it outside one's workgroup as an organization has multi business units, so docketing and archiving reuse across multiple business units becomes infeasible [24].
  A corporate-wide forum is needed to identify a product development cycle where reuse concerns can always be raised and resolved [18]
- **Economic barriers:** supporting corporate-wide reusable assets demands economic investment, particularly if reuse groups need a huge investment [24].
  The costs and benefits must be understood for a product life cycle based on a "Design for Reuse" philosophy. Reusable work-products must be viewed as capital assets [18].
- **Technical barriers:** Proper mechanisms are needed to ensure that guidelines, techniques, and standards for making things reusable are developed and followed [18].
- **Legal barriers:** Negotiations must be undertaken to determine how to retain rights to components developed under customer contract and recover costs in a reuse context. Mechanisms will be needed for payment and collection of royalties for use and reuse in the commercial arena [18].
- **Psychological barriers**: highly talented programmers are against reuse, as they believe in developing everything their way and reuse causes a not made her kind of an attitude [24].

4.5 Does Study Propose Any Maturity Model for Reuse?

Only 42% of studies proposed maturity model of reusability. Six papers presented models for measuring the maturity of reusability in organization. This let us extract five reuse maturity models:

- **RMM**: In 1991, Koltun and Hudson [29] presented the first version of the Reuse Maturity Model (RMM). The model, in fact, provides a concise form of obtaining information on reuse practices in organizations. The model is composed of five levels, and ten dimensions or aspects of reuse maturity were enumerated.
- **RCM:** In 1993, Davis [30] presented the Reuse Capability Model (RCM), an evolution of the STARS' reuse maturity model. RCM aids in the evaluation and planning for improvements in the organization's reuse capability. The reuse adoption process is a solution to implement a reuse program and it is based on the implementation model defined by [31].
- **RRM:** Another reference model for software reuse called Reuse Reference Model (RRM) was presented by [32]. RRM incorporates both technical and organizational elements that can be applied to establish a successful practice of software reuse in the organization.
  Based on the research results and case studies, Rine and Nada conclude that the level of reuse, as defined in RRM, determines the capability of improvements in the productivity, quality and time-to-market of the organization.
- **RISE:** The specification of the initial RiSE Maturity Model was described by [11]. It presented the approach for creating the model, its current structure and its levels.
  The main purpose of RiSE Maturity Model is to support organizations in improving their software development processes. In particular, the model has to serve as a roadmap for software reuse adoption and implementation.
  The RiSE Maturity Model consists of the following elements: Maturity Levels (Ad hoc Reuse, Basic Reuse, Initial Reuse, Organized Reuse, and Systematic Reuse), Goals assigned to each level, Perspectives (Organizational, Business, Technological and Processes) and Practices grouped in levels and perspectives.
- **RCMM:** RCMM: Reuse Capability Maturity Model, was presented by[18] as a maturity

model with focus on reuse and describes which are basic in order to ensure a well-planned and controlled reuse oriented software development.

In RCMM, there are five levels inspired by the famous SEI's (Software Engineering Institute) Capability Maturity Model. Each level represents a stage in the evolution to a mature reuse process. A set of maturity goals for each level and the activities, task and responsibilities.

4.6 Does Study Propose any Attributes that Affect the Software Reusability?

Most of papers point out a set of attributes that affect the reuse.

Literature survey reveals common attributes that are believed to influence reusability of software components. In particular, papers [4], [6], [14], [20], [23], [25], emphasis on the following nine attributes:

- **Understandability:** A software component is more usable if it is can be easily understood. So when modules are well documented then their understandability is high i.e. new developers understand easier code of module having comment lines.
- **Portability:** It is the ability of a component to be transferred from one environment to another with little modification, if required. If a component has little or no portability then its chances of being reused reduce.
- **Maintainability:** The degree to which the system or module of the software can be modified easily in order to fix bugs, adding quality attributes, or for adjustment of the operating environment change, increase efficiency of the system.
- **Adaptability:** Adaptability determines as how easily software satisfies requirement or and user requires of the new environments from being system and system constraints.
- **Interface Complexity:** Complex interfaces will lead to the high efforts for understanding and customizing the components. Therefore, for better reusability, interface complexity should be as low as possible.
- **Flexibility:** Flexibility is the ability to use a software component in multiple configuration. To use some source code component, it should be flexible to be used in many contexts.
- **Stability:** Stable means the reasonability error is free and it may be adapted with confidence that there is no bug.
- **Independence**: This attributes refers to the property of a component or software to perform its tasks by itself. More is the independence of software more will be reusability, otherwise its dependability makes it difficult to be used again and again.
- **Documentation:** Documentation is intended to make software components easier to understand.

4. CONCLUSION

Although software reusability can significantly improve productivity and quality of a software product, it is considered as difficult task especially for legacy software.

In this study, we presented a literature review of the most up-to-date research work published on software reusability. This review of various software reusability concepts offers a good understanding of reusability for accelerating the adoption of reusability in software development.

We found in this study that few works examined barriers of reusability, which can motivate organizations to adapt software reusability approaches. Also the studies about maturity models of software reuse are limited, so exploring this domain for helping organizations to audit his maturity reuse levels, can be a subject of a future work.

REFRENCES:

[1] N. S. Gill, S. Sikka, "Inheritance Hierarchy Based Reuse & Reusability Metrics in OOSD", International Journal on Computer Science and Engineering (IJCSE), Vol. 3, n. 6, 201, pp. 2300-2309.

[2] B. William, Frakes and Kyo Kang, "Software Reuse Research: Status and Future", IEEE Transactions on software engineering, Vol. 31, n. 7, 2005.

[3] B. Jalendar, A. Govardhan and R. Emchand, "Desiging code level reusable software components", International Journal of Software Engineering & Applications, Vol. 3, n. 1, January 2012, pp. 219-229.

[4] Y. Singh, P. K. Bhatia, O. Sangwan1, "software reusability assessment using soft computing techniques", ACM SIGSOFT Software Engineering Notes, Vol. 36, n. 1, January 2011, pp. 1-7.

[5] K. Kaur, N. Mohan and Dr. P. S. Sandhu, "Reusability of Software Components using J48 Decision Tree", Proceedings of the International Conference on Artificial Intelligence and Embedded Systems, 2012, pp.69-71.

[6] Y. Y. Ibraheem, A. M. Abualkishik and M. Z. Mohd Yussof, Taxonomy, "Definition, Approaches, Benefits, Reusability Levels, Factors and adaptation of Software Reusability: A Review of the Research Literature", Journal of Applied Sciences, Vol. 14, n. 20, 2014, pp. 2396-2421.

[7] B. Kitchenham, S. Charters, "Guidelines for performing systematic literature reviews in software engineering", EBSE, 2007.

[8] E. S. Almeida, A. Alvaro, and D. Lucrédio, V. C. Garcia, and, S. R. L. Meira, "RiSE Project: Towards a Robust Framework for Software Reus", Proceedings of the In IEEE International Conference on Information Reuse and Integration (IRI), 2004, pp. 48–53.

[9] O. P. Rotaru, M. Dobre, "Reusability Metrics for Software Components", Proceedings of the the 3rd ACS/IEEE International Conference, 2005, pp. 24.

[10] J. Finnigan, J.Blanchette, "A Forward-Looking Software Reuse Strategy", Proceedings of the IEEE Aerospace Conference, 2007, pp. 1-9.

[11] V. Garcia, D. Lucrédio, A. Alvaro, "Towards a Maturity Model for a Reuse Incremental Adoption", Proceedings of Simpósio Brasileiro de Componentes, Arquitetura e Reutilização de Software (SBCARS), 2007, pp. 61-74.

[12] F. McCarey, M.O. Cinneide and N. Kushmerick, "Knowledge reuse for software reuse", Web Intelligence and Agent Systems, Vol. 6, n. 1, 2008, pp. 59-81.

[13] R. Kamalraj, B.G Geetha, G. Singaravel, "Reducing efforts on software project management using software package reusability", Proceedings of the IEE Advance Computing Conference, 2009, pp. 1624-1627.

[14] A.Sharma, P.S. Grover and R. Kumar, "Reusability assessment for software components", ACM SIGSOFT Software Engineering Notes, Vol. 34, n. 2, 2009, pp. 1-6.

[15] M. Dinsoreanu, I. Ignat, "A Value Analysis Model for Measuring Software Reuse", Proceedings of the Second International Conference IEE of Applications of Digital Information and Web Technologies(ICADIWT'09), 2009, pp. 846-848.

[16] P.S. Sandhu, P. Kakkar, S. Sharma, "A survey on software reusability", Proceedings of the Second International Conference IEE of Applications of Mechanical and Electrical Technology (ICMET), 2010, pp. 769-773.

[17] G. Singaravel, V. Palanisamy, A. Krishnan, "Overview analysis of reusability metrics in software development for risk reduction", Proceedings of the International Conference IEE of Innovative Computing Technologies (ICICT), 2010, pp. 1-5.

[18] K.S Jasmine, R. Vasantha, "A New Capability Maturity Model For Reuse Based Software Development process", IACSIT International Journal of Engineering and Technology, Vol. 2, n. 1, February 2010, pp. 112-116.

[19] W. Spoelstra, M. Iacob, M. Sinderen, "Software Reuse in Agile Development Organizations - A Conceptual Management Tool", Proceedings of the 2011 ACM Symposium on Applied Computing, 2011, pp. 315-322.

[20] B.Jalender, A. Govardhan,R. Emchand, "Designing code level reusable software components", International Journal of Software Engineering & Applications (IJSEA), Vol. 3, n. 1, January 2012, pp. 219-229.

[21] B. Koteska, G. Velinov, "Component-Based Development: A Unified Model Of Reusability Metrics", Proceedings of ICT Innovations 2012: Secure and Intelligent Systems, 2013, pp. 335.

[22] C. Hong, "Minimal information for reusable scientific software", Proceedings of the 2nd Workshop on Working towards Sustainable Scientific Software: Practice and Experience, 2014.

[23] S. Thakral, S. Sagar and Vinay, "Reusability in Component Based Software Development - A Review", World Applied Sciences Journal, Vol. 31, n. 12, 2014, pp. 2068-2072.

[24] R. Keswani, S. Joshi, A. Jatain, "Software Reuse in Practice", Proceedings of the IEEE International Conference on Advanced Computing & Communication Technologies (ACCT), 2014,pp. 159-162.

[25] C. Monga, A. Jatain, D. Gaur, "Impact of Quality Attributes on Software Reusability and Metrics to assess these Attributes", Proceedings of the IEEE International on Advance Computing Conference (IACC), 2014, pp. 1430-1434.

[26] Md. Iftekharul A. Efat, Md. S. Siddik, M. Shoyaib, S. M. Khaled, "Feature Prioritization for Analyzing and Enhancing Software

Reusability", Proceedings of the IEEE International Conference on Informatics, Electronics & Vision (ICIEV), 2014 pp. 1-5.

[27] P. K. Singh, O. P Sangwan, A. P. Singh, A. Pratap, A Framework for Assessing the Software Reusability using Fuzzy Logic Approach for Aspect Oriented Software, International Journal of Information Technology and Computer Science, Vol. 7, n. 1, 2015, pp. 67-72.

[28] P. Clements and L. Northrop, Software Product Lines: Practices and Patterns, Addison-Wesley, 2001.

[29] P. Koltun, P. A. Hudson, "A reuse maturity model", Proceedings of the 4th Annual Workshop on Software Reuse, 1991.

[30] Davis, Ted, "The reuse capability model: A basis for improving an organization's reuse capability", Proceedings of 2nd ACM/IEEE International Workshop on Software Reusability, 1993, pp. 126-133.

[31] R. Prieto-D´ıaz, "Making software reuse work: An implementation model", ACM SIGSOFT Software Engineering Notes, Vol. 16, n. 3, 1991, pp. 61-68.

[32] D.C. Rine, N. Nada, An empirical study of a software reuse reference model, Information and Software Technology, Vol. 42, n. 1, 2000, pp. 47-65.