Volume 2, Issue 5
August 2013

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X

Using Open Source Software in Reuse-Intensive Software Development – A Qualitative Study

[1]Fazal-e-Amin, [1]Ahmad Kamil Mahmood, [2]Aized Amin Soofi, [2]M. Irfan Khan

[1]Computer and Information Sciences Department, Universiti Teknologi PETRONAS,
Bandar Seri Iskandar, 31750 Tronoh, Perak, Malaysia.
[2]College of Computer Science and Information Studies
Government College University, Faisalabad, Pakistan
E-mail: fazal.e.amin@gmail.com, kamilmh@petronas.com.my, aizedamin@yahoo.com, softchannel2000@hotmail.com

ABSTRACT

Open Source Software (OSS) is one of the emerging areas in software engineering. Reuse of OSS is employed in reuse-intensive software development such as Component Based Software Development and Software Product Lines. OSS is gaining the interest of the software development community due to its enormous benefits. The context of this study is the use of OSS in reuse-intensive software development. The use of OSS in the systematic reuse of software, such as in Software Product Lines (SPLs) is a new phenomenon. The aim of this study is to identify the different dimensions of this phenomenon. In this study, a qualitative method, namely the interview, is used to acquire data. Interviews are conducted with seven respondents. The data is analyzed using an adapted form of grounded theory. The results of this study include seven categories and their 39 subcategories / dimensions. The results of the study are compared with contemporary studies in this area to highlight the contributions and to complement them. The findings of this study provide an in-depth view of the issues related to the use of OSS in reuse-intensive software development. These findings will help the community to improve their practices and to initiate steps to cope with the challenges.

Keywords: *Open source software, reuse-intensive software development, interview, grounded theory*

1. INTRODUCTION

Open Source Software (OSS) is gaining the interest of the software engineering community due to its numerous benefits. These benefits fall into different dimensions. One dimension is financial benefits, e.g. the reduction in maintenance cost [1] and the escape from vendor lock-in [1, 2]. Another dimension is technical benefits including having a large number of developers and testers [3, 4] and less maintenance risk [5]. Other dimensions include user support from the community [4], encouraging innovation [6, 7] and increased collaboration [8]. As we can see, there are multifaceted advantages to the use of OSS. The benefits may relate to social aspects or to financial ones. The factors which contribute to the popularity of OSS may also include increased bandwidth, improved search facilities, and the existence of code conjurers [9]. The growth of the Internet is also one of the factors which has a huge impact on the way that software is developed, marketed, and supported [10]. The use of OSS in Component Based Software Engineering is already a norm in the industry [11]. Recently, researchers have envisioned the use of OSS in SPL development [12, 13]. So, OSS is entering into a new arena. Currently available knowledge in this research area is limited. This lack of knowledge is also recognized in [14]: "there has been no systematic synthesis of the OSS challenges reported in the literature." Therefore, it is obvious that there is a need to explore the use of OSS in reuse-intensive software development, especially in Software Product Lines (SPLs). In this study, not only the challenges, but also other dimensions to using OSS are explored, such as the current reuse practices, the desirable characteristics of OSS and the use of OSS in SPLs.

2. RESEARCH METHODOLOGY

This is an objective exploratory study. The study comprises a literature search and interviews [15]. The literature contains information regarding the use of OSS in Component Based Software Development. Furthermore, the use of OSS in SPLs has recently been proposed by researchers. As regards the interviews, the respondents were selected carefully on the basis of their knowledge and expertise in this area. The analysis of the qualitative data gathered through the interviews was performed using the grounded theory approach [16]. The details on the analysis and results are presented in the next sections. The results are in the form of categories and sub-categories, which provide an insight into the different dimensions of the categories. The findings are a contribution towards the body of knowledge. Details of the interviews conducted in this study are provided in next section.

2.1 Interview

Qualitative evaluation and validation approaches seek to collect data in the form of text and pictures. The interview is one of the forms of qualitative data collection. In [17], a detailed discussion about the use of participant observation and interviewing is provided.

The interview is a means of collecting primary data; it is a conversation between two persons, one of whom is a researcher. Interviews can be used for data collection where the nature of the study is exploratory. Interviews are helpful when the data to be gathered is about a person's knowledge, preferences, attitude or values [18]. Interviews are useful in situations when the logical order of the questions is not clear or predetermined [18]. Interviews may help to gather

impressions and opinions about something [17]. Interviews enable one to get personalized data, provide an opportunity to probe, establish technical terms that can be understood by the interviewee, and facilitate mutual understanding. The interview provides an in-depth view for exploring the perspective of informants [18]. Interviews enable the researcher to understand the experiences of others. Several types of interviews are reported in the literature [19]. In this study the semi-structured type of interview is used.

Unstructured interviews are costly in terms of time and resources as they require a lot of time to conduct the interviews and to analyze the data. On the other hand, structured interviews are efficient, requiring less time and resources. However, structured interviews follow a set pattern that does not allow for a detailed exploration of the issues. Semi-structured interviews offer a compromise, making use of both open-ended and specific questions. This combination allows the researcher to explore the issues by collecting expected information using specific questions, and unforeseen information from open-ended questions. The semi-structured type of interview is used in this study.

2.2 Interview Guide

An interview guide helps the researcher in organizing the interview. The contents of an interview guide include the list of open-ended questions to be asked during the interview and notes to direct the interviewer in the desired direction. Like field notes, an interview guide is again confidential, i.e. it is not shown to the respondent. For novice interviewers it is usually difficult to conduct the interview and write notes at the same time. An audio recording of the interview provides a solution to this problem. The permission to audio tape the interview is essential; it is ethically binding on the researcher to inform the respondent that the conversation is being taped.

The interview guide contains open ended questions, or in other words the issues to be discussed. The conversation starts with a brief introduction by the interviewer. In our case, an introduction of the topics is not necessary because the topics are already known to the researchers. In fact some of the respondents are experts who are well known in the research community. The respondents answered the questions differently due to their varying knowledge and level of experience. They used examples (citing names of software) and referred to their talks with other researchers. The transcribed interviews are not presented here, neither are the names, the places or the events. The crux of the conversation and results are presented in the results section.

2.3 Respondents' Profiles

The research issues investigated in this study are of a specialized nature. Not everybody working in industry or academia is able to answer these questions. The respondents were chosen based on their expertise. It should be noted that the respondents have up to date information regarding the research in this area and industrial practices.

The interviews were audio recorded and transcribed prior to performing the analysis. The first respondent is a software engineering researcher and developer. He has experience related to human computer interaction application development.

The second respondent is a researcher having a doctorate degree in software engineering in the area of software product lines. He is an author of many publications, some of which are book chapters. His publications include those specifically targeting software product lines and related issues.

The third respondent is an expert in software reuse research, and has been authoring research papers on software reuse since the 1980's. He actively participates in research activities and is currently the editor of a publication in software engineering published by a prestigious body. He is currently serving as the principle software architect in a well known organization.

The fourth respondent started his career as a software engineer and had been promoted to software project manager during his career. He has managed several projects in the domains of accounts, student information service, examination systems and automation of small industries and NGOs.

The fifth respondent has worked in the domains of micro finance, accounts, medical laboratories, visa system, and billing.

The sixth respondent is working in a multinational software development company. He has experience of working in the education and health sectors. Currently, he is serving as software quality assurance engineer.

The seventh respondent is also associated with the software industry, working in a well reputed and nationally certified software company. He has been involved in developing software related to the project management domain.

The profiles of the respondents are diverse, which influenced the design of the interview guide and meant that not all of the questions were posed to all of the respondents. Table 1 summarizes the profiles of the respondents. Different means were used for conducting the interviews due to the location of the respondents. These are shown in Table 2.

2.4 Qualitative Analysis Coding Process

The coding process of grounded theory is used for the analysis part of this study. Following the grounded theory approach, the coding process starts with open coding [16]. It is the first analytical step that 'opens up' the transcribed interview. The concepts are identified and names are given to them. The open coding process results in a list of codes. In addition a word cloud is generated. This step is taken to make sure that none of the recurring words are missed. After the analysis of all the transcriptions, we have all of the concepts. Similar ones are grouped into categories. The categories are named such that they provide meaningful insight into them. After the identification of categories, the properties and dimensions are developed by answering the 'when', 'where', 'why', and 'how' questions about the categories. The main categories are divided into subcategories according to the different properties and dimensions of the main categories.

The categories and respective sub-categories are linked during the axial coding process [16]. Axial coding is the process of reconnecting the data. The linking of categories and subcategories provides more information and dimensions of the phenomenon. The category, itself, is a phenomenon; on the other hand, a sub category is not a phenomenon by itself, but it answers questions about a phenomenon.

Volume 2, Issue 5
August 2013

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X

Selective coding [16] is the process of integrating and refining the categories which are identified in the axial coding process. During selective coding, the researcher reaches a point where no new property, relationship or dimension emerges. This point is termed theoretical saturation [16]. A pictorial depiction of the coding process is presented in Figure 1. The qualitative data was analysed using the 'atlas.ti' tool [20].

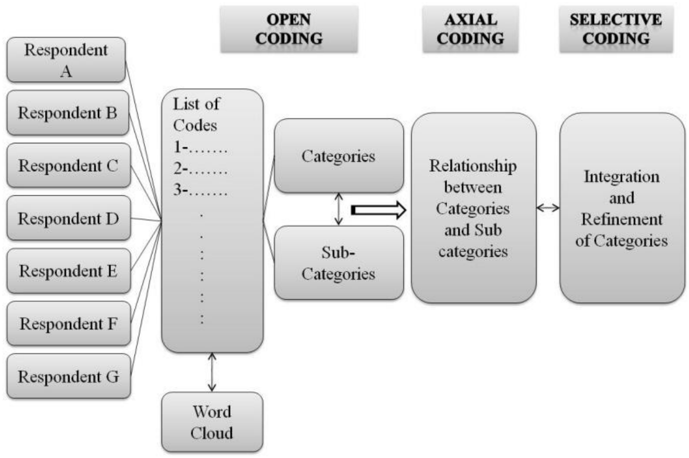

Figure 1 Coding process

Table 1 Respondents' profiles

Respondent ID	Experience	Experience Type	Current Affiliation
Rsp-A	5 years	Academic, Software Industry	Academia
Rsp-B	10 years	Academic, Software Industry	Industry
Rsp-C	22 years	Academic, Software Industry	Industry
Rsp-D	8 years	Academic, Software Industry	Academia
Rsp-E	10 years	Academic, Software Industry	Academia
Rsp-F	3 years	Software Industry	Industry
Rsp-G	4 years	Software Industry	Industry

Table 2 Means used to conduct interviews

Means used	Skype	Face to face	Telephone	Total
Number of Interviews	3	3	1	7

2.5 Word Cloud

A world cloud is a technique used to represent the frequencies of words in textual data. On the World Wide Web, word clouds are also referred to as tag clouds. Word clouds are used to depict the relative importance, frequency, and popularity of a word [21]. In this research, a word cloud is used in addition to 'open coding' to make sure that none of the recurring words are missed. The interview transcripts contain 6,483 words. A word cloud of these words is generated by an online word cloud service (www.tagxedo.com), and is shown in Figure 2. The cloud includes the 300 most frequently recurring words in the transcripts. The word cloud helped to ensure that the concepts related to these words are included in the code list.

Figure 2 Word cloud generated from interview transcripts

3. RESULTS

The findings of the interviews and literature search are presented in this section. These findings emerged into seven categories. The names and descriptions of these categories are provided in Table 3.

Table 3 Categories and their descriptions

Category ID	Category Name	Description
Cat-1	Challenges of OSS	These challenges are wide-ranging: the customer's viewpoint; the end user's viewpoint; commercial and secure application development issues.
Cat-2	Current reuse practices	Knowledge about current reuse practices as employed in industry are combined in this category. This knowledge is based on the experience of the respondents.
Cat-3	Using OSS in SPL	The views of the respondents on the use of open source software in product lines are put together in this category.
Cat-4	Role of OSS in promoting reuse	This category is based on the role of OSS in the promotion of reuse, i.e. why OSS is influencing reuse intense software development.
Cat-5	Factors affecting reusability	The factors of reusability are assembled under this category.
Cat-6	Desirable characteristics of OSS	The desirable characteristics of OSS, identified during the study, are presented in this category.
Cat-7	Suggestions	The suggestions provided by the respondents are presented in this category.

Table 4 Sub categories of 'Challenges of OSS'

Cat-1	Challenges of OSS	
Sub Category ID	**Sub Category Name**	**Representative Quote**
SC-1-1	Finding OSS	"Finding an OSS component is one of the challenges."
SC-1-2	Evaluating OSS	"If I find a required OSS component then its evaluation is a challenge."
SC-1-3	Lack of documentation	"….without proper information it is difficult to understand it." "If there is no proper documentation then other cannot understand the software neither can change nor modify it." "The challenge in the context of the open source is analyzing, usually OS comes along with source code without many documentation. So, it is very difficult to analyze without documentation."
SC-1-4	Reluctance for developers to make their software OSS	"The managers and owners of the company want to develop such type of product that they have ..... They don't want to contribute to the open source because they want to run their software house ..." "they are willing to use the OS but not to contribute to the OS because of their limitation and because of the market competition" "They don't want to contribute to the open source because they want to run their software house."

ITEE Journal
Information Technology & Electrical Engineering

ISSN: - 2306-708X

Table 5 Sub categories of 'Challenges of OSS' (cont.)

Cat-1	Challenges of OSS (cont.)	
Sub Category ID	Sub Category Name	Representative Quote
SC-1-5	Lack of information about intellectual property rights /copyright	"The developers have lack of information about the intellectual property rights in OSS, so they are afraid to share code." "Lineage of the software ensuring that no intellectual property so that is the biggest hesitation."
SC-1-6	Lack of adherence to coding convention /standard	"There should be some specific rules, common rules for each for the whole developer community or those contributing to the OSS." "If some immature developer is developing the software defiantly the code would be different from the professional developer."
SC-1-7	Security	"Any secure system which included OS but still there could be certain measures if the OSS. They did a scan on the source code rather than they incorporated as binary and they could do the necessary analysis to know that the OSS does not have any malicious code, entered in the software."
SC-1-8	Improper reviewing /comments	"I have seen some customers/users of OSS review but the main problem is there are no rules for writing a review, every reviewer is writing the review in their own context in their own way."
SC-1-9	Fear of losing market share	"If they develop a tool or software and they contribute of float it as OSS there are chances that they can't further work/earn."

3.1 Challenges of OSS

In this study, the challenges of OSS have been identified through a qualitative method. In this section these are presented, and categorized on the basis of the opinions of the respondents. These challenges fall into different dimensions. The list of challenges (sub categories) and their corresponding representative quotes are presented in Tables 4 and 5.

These challenges include finding and evaluating OSS, the lack of documentation, and the developers' reluctance to make their software OSS. Another challenge is that developers do not have appropriate information about intellectual property rights / copyright. A further challenge is the lack of adherence to coding standards. The security of OSS is also one of the major challenges. In relation to the challenge of finding OSS, improper reviewing and comments is one of the issues that needs attention. At the organizational level there is a fear of losing market share, which poses a challenge for the OSS community, as it deters developers from making their software open source.

A study [14] has also discussed the challenges of OSS. The authors made an argument that "there has been no systematic synthesis of the OSS challenges reported in the literature". However, their study was based on a literature survey. In the case of this study, the findings are based on interviews with the experts, researchers and practitioners together with a influence of what the literature says.

The common findings of this study and [14] are the challenges of finding and evaluating components, poor documentation, and the legal aspects such as copyright and intellectual property rights. The findings presented in this study are based on the viewpoint of users of OSS components, i.e. the software engineers. In this regard, the prominent findings are the issues of security, fear of losing market share (at the organizational level) and fear of losing one's job (at the individual software engineer's level). In a recent focus group study it was reported that security and documentation are among the most important technical factors [22]. These factors should be taken into account when selecting an OSS component.

3.1.1 SC-1-1 Finding OSS

The very first challenge in OSS is searching for it. Searching facilities are improving with the emergence of new search engines. Furthermore, enormous contributions are being made by numerous software engineers. The availability/accessibility of OSS has improved but it is still a challenge to find specific OSS. One of the reasons is that different cataloguing standards are employed by search engines. This lack of a standard makes it difficult for a new user (i.e. a software engineer) to search for a component using different search engines.

3.1.2 SC-1-2 Evaluating OSS

The evaluation of the OSS is another challenge. For example, in the first step (Finding OSS) when a required component is found, then the decision whether to use it or not is related to the evaluation of the OSS. The practices to evaluate OSS differ in different organizations. In small organizations the evaluation of OSS prior to using it is at the discretion of the software engineer. In such environments this evaluation depends on the knowledge and expertise of the software engineer.

3.1.3 SC-1-3 Lack of Documentation

Lack of documentation is related to the understandability of software. Lack of documentation affects the understandability/analysis of the software. So, without

having appropriate documentation it is difficult for software engineers to use it. One of the reasons for this issue is that a large number of developers contribute to many OSS components, and this complicates the provision of documentation.

The respondents consider documentation as the most important quality of OSS. The quality of documentation reflects the quality of the software. Another aspect of documentation is that it provides a record of the changes made to the OSS, i.e. it gives its history.

3.1.4 SC-1-4 Reluctance for Developers to Make Their Software OSS

In industry, managers encourage open source based development. However, sometimes it is a one way interaction, meaning that open source is used but is not contributed to. The owners/managers of software houses want to develop software using open source components to save time and money, however, they want to capture market share and that is why they force software engineers not to share code. Sometimes software engineers (developers) want to contribute to the open source community, but they are constrained by their owners. In such a situation, software engineers are forced to have a one way interaction with OSS and are unable to contribute to OSS, and have to keep their innovations to themselves. Fear of losing their job is also associated with the reluctance of software engineers. They perceive OSS contribution as a threat to their job.

3.1.5 SC-1-5 Lack of Information about Intellectual Property Rights/Copyright

Software developers are not fully conversant with the intellectual property rights and licensing information of open source software. There are issues with legacy code and the 'lineage' of software. As an example, a software engineer might work for an organization and then leave that organization. The software engineer might retain the application code. The individual may join another organization and start using the code or he/she may make it available as OSS.

3.1.6 SC-1-6 Lack of Adherence to Coding Convention/Standard

There is a lack of adherence to coding conventions. This reduces the understandability of the code. The level of expertise of the programmer and type of experience that they have had also plays an important role. The respondents suggested that OSS contributors should follow coding conventions. A check should be undertaken of the code, and only if it meets the standard should it be included in an open source portal. Without this check, the code's worth is questionable. Instead of spending large amounts of time understanding others code, software engineers would prefer to write their own.

3.1.7 SC-1-7 Security

There is a security concern when using OSS in critical and highly secure application domains, such as the defence, government and financial sectors. In such situations there should be mechanisms for code scanning to ensure that the code is clean, meaning that there is nothing malicious in the code.

The issue of security is of great importance. In a recent focus group study [22], the security of components has been identified as an important technical factor that influences the selection of components. Currently, software engineering researchers are working on empirical studies on open source and closed source software, see [23] for example. This study concludes that there is no significant difference in both open and closed source software development in terms of security.

3.1.8 SC-1-8 Improper Reviewing/Comments

Reviewer/user comments about OSS play an important role for the potential user of OSS. One can learn about the software prior to downloading and using it. There is a huge amount of code (software) available over the Internet. Users have written comments about some of it. However, the issue is that there is no standard for reviewing code and writing comments about it. The person who writes a comment shares a personal experience with a particular piece of software. Sometimes, the context of its use is not clear, which raises questions in the reader's mind. It is suggested by the respondents that there should be standards for writing comments about software so that potential OSS users can easily extract the required information. The challenge of improper reviewing and comments can be related to the challenge of several descriptions of the same document, which is identified in [24].

3.1.9 SC-1-9 Fear of Losing Market Share

This issue is more relevant to product line and domain based software development, where companies target a specific domain and group of potential customers. In such situations the software organizations may use OSS but do not want to contribute software to an OSS repository because they want to keep their innovations to themselves. In this way they sustain themselves in a particular domain. There is a fear associated with sharing - if they share code they will risk losing their position in the market.

3.2 Current Reuse Practices

The current reuse practices are identified in this study. The findings which fall under this category are related to the knowledge about current reuse practices as employed in industry. This knowledge is based on the experience of the respondents. The list of sub categories is presented in

Table Table 6, along with their corresponding representative quotes. The current reuse practices include knowledge reuse, looking at the demonstration of software and the initialization of a product.

In [25] and [26], it is stated that OSS developers reuse existing code in three forms. These three forms include reuse of components, single lines of code, or algorithms / methods. In this study, it is identified that another form of reuse is the translation of logic from one programming language to another. There is the situation where a developer searches for a specific code in a specific language but she/he can only find the functionality in some other language. Then

she/he tries to translate it in the desired language. In other words, creates a replica of the program in the other language. Demonstration versions of OSS are used to assess the effort required to modify code or the number of modifications to be made. Furthermore, the modules where the changes are required can be easily identified by using a demonstration version.

3.2.1 SC-2-1 Knowledge Reuse

There is a form of reuse in which knowledge is reused. Imagine a situation where a software engineer is searching for a component written in one language but can only find it written in another language. The software engineer can reuse the knowledge inherent in the logic but rewrite the code. This form of reuse is helpful where the bulk of the logic remains the same and the objectives of reuse are achieved with some changes / adaptation. A demonstration version of software is also related to knowledge reuse, as explained in the next section.

3.2.2 SC-2-2 Demo

Much OSS comes with a demonstration version; this is very helpful for understanding the software. The potential user of OSS can base his/her decision on whether or not to use the component on the success of using the demonstration version. In other cases, the software engineer wishes to use a component following some modifications. A demonstration aids the software engineer in thinking how best to make the modifications. Software engineers can sometimes gain a better idea of the functionality of a part through using a demonstration rather than studying a large amount of code. In this way target areas of the program requiring modification can sometimes be more easily identified.

3.2.3 SC-2-3 Not Started From Scratch

There is an opinion that starting a product 'from scratch' is impractical. Software engineers start developing with having some components at hand. It saves time and other resources. At a higher level the same goes for software product lines. Product lines are started after having prior knowledge and an awareness of their applications in the domain. Reuse as an aid to getting started can facilitate a competitive advantage, allowing a new software product to be developed in a short period of time.

The association of an organization with a specific domain helps to develop trust in its new products. The customer prefers software products from companies that are already established in a specific domain.

Table 6 Sub categories of 'Current reuse practices'

Cat-2	Current reuse practices	
Sub Category ID	Sub Category Name	Representative Quote
SC-2-1	Knowledge reuse	"We can just take an idea and we can reuse the idea."
SC-2-2	Demo	"Demo could be helpful for programmers they can understand the software by a demo of the software, demo gives an idea to the programmer…"
SC-2-3	Not started from scratch	"Now I think this concept that you start developing a product from scratch is impractical because right now the time is scarce in the world." "Software product lines are started with having some component in hand means the company is already working in this domain and that is why OSS may help them to start product line or to add new product into the line."

Table 7 Sub categories of 'Using OSS in SPL'

Cat-3	Using OSS in SPL	
Sub Category ID	Sub Category Name	Representative Quote
SC-3-1	Fast transition	"If such setup is developed (using open source to build family of systems) then the transition from manual to computerize will be much faster."
SC-3-2	OSS is attracting SPL community	"OSS is very attractive to product line community."
SC-3-3	Improvement in quality	"The number of times a component is reused its quality is improved." "Reuse also refines the product."
SC-3-4	Provides opportunities	"It's a good opportunity that you have idea or free code and then you

develop product lines."

Table 8 Sub categories of 'Role of OSS in promoting reuse'

Cat-4	Role of OSS in promoting reuse	
Sub Category ID	Sub Category Name	Representative Quote
SC-4-1	Saves time	"…it saved time otherwise it will take weeks to develop it."
SC-4-2	Less effort required	"It saves around 80% of the time and less effort is required."
SC-4-3	Ease of development	"Nowadays open source development is encouraged within the software development community, and people are going towards open source, they feel very easy to develop."
SC-4-4	Market trust	"The company which reuse can develop a system in months and other may take a year… the company which is running a product line in some domain it also develop a trust."

3.3 Using OSS in SPL

This category is based on the views of the respondents on the use of open source software in product lines. The sub categories are presented in Table 7 with their corresponding representative quotes.

The use of OSS in an SPL facilitates a fast transition towards automation and the entering into new markets. OSS is a platform which provides components; it attracts the SPL community. The attraction is due to the benefits of OSS. The use of OSS in an SPL improves the quality of the software. OSS is an opportunity for the SPL community to add more innovations to their product lines.

3.3.1 SC-3-1 Fast Transition

The use of OSS to develop a product line provides an opportunity to develop a new product in less time. More generally, it hastens the transition of manual to automated systems as many of the systems which we interact with in our daily life are similar to one another. SPLs deal with such similarities (commonalties). The infrastructure of OSS core assets can be used to initialize many specialized products.

3.3.2 SC-3-2 OSS is Attracting SPL Community

OSS is attracting the product line community in the sense of starting a new product. An organization can develop a new product in less time using OSS. On the other hand, product lines are seldom started from scratch. So, OSS provides a good start to the SPL community. Obviously the standards and quality of OSS is an issue in the case of OSS based SPLs.

3.3.3 SC-3-3 Improvement in Quality

The reuse of software improves its quality. However, in the case of SPLs it is even more beneficial. The core assets are reused in multiple applications; this reuse refines the components. The more times a component is reused the greater is its quality. Another aspect of this is that the end user/customer is in a better position to state his/her requirements and comment on a system after using a similar one. The findings are in line with the other available studies. The improvement in quality that OSS brings about can be credited to fewer defects per line of code [27], and to improved reliability [28, 29].

3.3.4 SC-3-4 Provides Opportunities

The respondents considered OSS based SPLs as a window of opportunity. This reuse may lead to the inter-organizational reuse of the components. Reuse will be moved to a higher level, one that finds commonalties among domains. Such types of core assets will be developed which are for use by multiple domains.

3.4 Role of OSS in promoting reuse

This category is based on the role of OSS in the promotion of reuse, i.e. why OSS is influencing reuse intense software development. The sub categories and their corresponding representative quotes are presented in Table 8.

The role of OSS in promoting reuse has four dimensions. These dimensions include time and effort saving aspects, ease of development and market trust. The market trust, or in other words the trust of the customers, is gained by an organization entering into new domains by using OSS.

3.4.1 SC-4-1 Saves Time

The topmost benefit of reuse is that it saves time. The results of a survey based study [30] state that by reusing OSS the developer can save time for other important tasks of the project. Software engineers prefer to reuse a component if one is already available. On the organizational level and in product line practices reuse results in a short delivery time. New products can be launched in a shorter time. It provides a

competitive edge and there is time available for experimentation and innovation to enhance the features of the product.

3.4.2 SC-4-2 Less Effort Required

Reuse also results in saving effort. Less effort is required to develop software using OSS as compared to starting from scratch. In the context of SPL, product lines are seldom started from scratch. The organizations which move towards the development of an SPL already have experience of that particular domain.

3.4.3 SC-4-3 Ease of Development

OSS allows for easier development; that is why its use is encouraged in the software industry. Developing a new product is easy. A developer, who is new to the domain, can get domain knowledge by reusing the components.

3.4.4 SC-4-4 Market Trust

The reuse of OSS enables a company to launch a product more rapidly. A potential customer prefers products of a company that is already developing software in that particular domain. OSS usage develops the trust of the customer.

3.5 Factors Affecting Reusability

The factors identified as the attributes of reusability are assembled under this category. The factors and corresponding quotes are presented in Table 9.

The findings reported in this study have extended the body of knowledge by adding new attributes of reusability. The salient feature of this study is the identification of the reusability attributes from the perspective of the software developer. This identification is based on interviews. The use of this method of enquiry is motivated by another view of reusability. This view is presented in [31]; it states that reusability is a form of usability from the perspective of the software developer. Using an interview is one of the most suitable methods in such a situation. In a recent focus group study it was reported that documentation (as we mentioned earlier) and maintainability are among the most important technical factors [22]. These factors should be taken into account when selecting an OSS component.

3.5.1 SC-5-1 Flexibility

Flexibility is related to reusability in two ways. First, it is the ability of a component to be used in multiple configurations. Second, it is a necessary attribute concerning future requirements and enhancements.

3.5.2 SC-5-2 Maintainability

Maintainability is related to reuse in terms of error tracking and debugging. If the component is maintainable it is more likely to be reused. In cases where OSS components are running on systems connected to another system then a bug is particularly problematic. Sometimes debugging a component on one configuration may not work on other configurations. On the other hand, in black box reuse, maintainability is not considered a factor of reusability.

3.5.3 SC-5-3 Portability

Portability is considered a factor in the sense that a cohesive component is more portable. A component having all the necessary information within it or having less interaction with another module during its execution is more reusable. Again in the case of black box reuse it is not a factor.

3.5.4 SC-5-4 Scope Coverage

Another characteristic of open source components is the extent of their scope. A developer would prefer a component to cover as much of the application's functionality as possible. Large components are of concern as it often means a high level of complexity and poor understandability. Furthermore, scope coverage is important in situations where future enhancements have already been envisioned, or where there is the likelihood that more features will be added in the future.

3.5.5 SC-5-5 Stability

Stability is one of the identified factors of reusability in this study. The respondents regard stability as an important factor to be considered while making decisions. Stability of a component refers to its quality of being error free. In [22] the concept of stability is linked to the "ad hoc standard." The explanation of the term 'ad hoc standard' is stated as "that they are used in many products of that kind". Our notion to explain this phenomenon is 'safety in numbers'. OSS contributed to by many developers and used in many applications is more stable. Stability is also related to the usage history of the component.

3.5.6 SC-5-6 Understandability

The respondents also have a consensus of opinion on the importance of the understandability attribute. It is also related to the maintainability of the component; a component that is easy to understand is easy to maintain. Understandability affects the reliability of a component.

3.5.7 SC-5-7 Usage History

In this study, 'usage history' is identified as one of the factors that influence reusability. Usage history provides a hint about the usefulness of the component. Another side of usage history is the maturity of the component. The component can be considered mature if it is used in many applications. The use of a component in many applications also reflects its quality of interoperability. It provides confidence to the potential user that a component can be easily adapted. Another aspect of usage history is that the use of a particular OSS in different applications provides an example of usage of the component. This example can be effective for learning purposes. A similar concept to 'usage history' is 'release history', as mentioned in [32]. Release history refers to "how often the new releases come out".

3.5.8 SC-5-8 Variability

Variability is one of the factors identified; increased variability decreases understand-ability. Variability is also seen as the configurability of a component, that it can be configured in multiple configurations. Variability is also related to the scalability property of a component, that is it can

be scaled up whenever required. Variability is further examined and explained in [33], metrics to measure variability at source code level are presented in [34].

Table 9 Identified factors and representative quotes

Cat-5	Factors Affecting Reusability	
Sub Category ID	Sub Category Name	Representative Quote
SC-5-1	Flexibility	"Flexibility refers to the ability to use it in multiple configurations." "In order to reuse some component source code it should be flexible enough to be used in several contexts." "Flexibility is necessary because there are changes required with the passage of time, so it saves you not to be bound."
SC-5-2	Maintainability	"Maintainability is a large problem in such situations when you use OSS and we are running the system with connectivity with other systems; so every time there are some bugs and removing the bugs in other code that is developed by some else is very difficult for the developer."
SC-5-3	Portability	"Portability is also related to the install ability, it should be taken care and portability should be economical we don't have to install other software to run a component in other systems."
SC-5-4	Scope Coverage	"That depends on the situation but normally we choose the more coverage component as compared to the less covered one." "… it depends on the application if we want to extend further our application then we will go for more features."
SC-5-5	Stability	"Stable meaning reasonably error free and it could be used with confidence that there is no bug."
SC-5-6	Understandability	"If I don't understand it then I can't show that it is reliable and prove it to myself then I am not going to use it." "Size can be managed but if it is not understandable then it is difficult to reuse"
SC-5-7	Usage History	"Usage history also shows the maturity of the component and how many people have used and made changes to it." "In many cases open source software is used by many people many engineers, already proven its usefulness."
SC-5-8	Variability	"Variability is a two edge sword in other words there are advantages and disadvantages."
SC-5-9	Documentation	"If there is lack of documentation then I mean it creates hurdles to understand the code for any other developer or the software engineer." "If there is no proper documentation then others cannot understand the software neither can change nor modify it."

3.5.9 SC-5-9 Documentation

The respondents consider documentation as one of the most important factors affecting flexibility, understandability and reusability. The issue of documentation is multifaceted. Usually, OSS comes without much documentation. OSS is developed and contributed to by many developers. The number of developers may reach up to the thousands, as in the case of Linux. The code size increases rapidly. So, it is very difficult to analyze code without documentation.

Documentation is associated with understandability. The lack of documentation, or poorly maintained documentation, hinders understandability. Documentation also provides a record of the component; the component history can be known by seeing the documentation. [32] recommends considering the documentation of OSS as a criterion for selecting candidate OSS. The finding presented in this study

that documentation is one of the factors which affect reusability is in line with [32].

3.6 Desirable Characteristics of OSS

The desirable characteristics of OSS identified during the study are presented in this category, and shown in Tables 10. In this study, the desirable characteristics of OSS are seen from the perspective of academics and industry.

3.6.1 SC-6-1 Academic Perspective

The desirable characteristics of OSS, from an academic perspective, include the availability of test cases associated with the open source software. This finding is in line with [22] where the availability of test cases is considered as one of the plus points. The primary focus of an OSS developed in academia is innovation and functionality. OSS in academic settings is intended to extend the body of knowledge. There is room for experimentation in academia.

3.6.2 SC-6-2 Industrial Perspective

A business desires different characteristics from software than academia does. Firstly, there is no room for experimentation in business environments or in commercial software development. The critical factor in a business environment is risk aversion. Several methods are used for the assessment and mitigation of potential risks.

3.6.3 SC-6-3 Maintenance Support

Maintenance is one of the issues in OSS. This is because of the shared/lack of ownership of the software. The potential user of OSS looks for its maintenance support. This factor is important as it influences the decision to use a particular OSS. OSS having maintenance support is preferable.

3.6.4 SC-6-4 Maintenance Agreement

In some situations companies may opt for a maintenance agreement with the developing organization. This kind of agreement covers extensions or changes to the software. On the one hand, the customer prefers maintenance agreements for their future needs or enhancements to the systems. On the other hand, software development companies earn additional revenues from these agreements.

3.6.5 SC-6-5 Infrastructure Support

Infrastructure support is identified as one of the desirable characteristic of OSS in this study. The range of infrastructures that support execution of the OSS is one of its defining characteristics. Here, the term 'infrastructure' refers to the operating system, web application server or the graphical user interface. So, OSS with more comprehensive infrastructure support is preferable. Wider infrastructure support makes the OSS an affordable choice under most circumstances. 'Infrastructure support' is also mentioned in [32], as one of the criteria for choosing OSS. It suggests asking the question: "Are they (OSS) compatible with the rest of your infrastructure?"[32].

3.6.6 SC-6-6 Maturity of OSS

The maturity of OSS is identified as a desirable characteristic in this study. The maturity of the OSS plays an important role in the selection making decision. One way to know the maturity of OSS is to look for its usage in different scenarios. These examples of use provide evidence to the potential user. The potential user may find similarities or differences in the examples and scenarios. The comparison helps him/her to build confidence in particular OSS. The potential user may identify related threats and risks. 'Maturity of OSS' is also identified as a desirable characteristic by [22] and [32], where the maturity of a community is considered to gauge the maturity of OSS.

3.6.7 SC-6-7 Error Handling Mechanism

The availability of an error handling mechanism is a desired characteristic of OSS. An error handling mechanism includes knowledge about the error types, related messages and their remedies. A remedy may include the types of parameters required to remove the error.

3.6.8 SC-6-8 Scalability

The capability of OSS to be scaled is considered a desirable characteristic. The scalability of OSS is its ability to handle the growing needs of the organization. It is also considered a variability parameter - that new functionality can be added or existing functionality extended.

3.7 Suggestions

The suggestions provided by the respondents are presented in this category, and are shown in Table 11.

3.7.1 SC-7-1 Inter Language Reuse

Inter language reuse of OSS is one of the suggestions. It can be viewed as a challenge to software engineering. Generic artifacts such as design documents and requirements specifications can be implemented in different languages. This is due to their abstract nature. However, code assets lack this level of abstractness. One of the possible solutions is the conversion of code of one language to another with the help of some intermediating software.

3.7.2 SC-7-2 Software Agents

The development of software agents is suggested by the respondents. These agents should be capable of guiding the software developer as to which kind of changes is required when adapting a particular component. These agents may help users by creating a meta-data file, containing the details of structures, classes, their types and relationships. This is so that the developer can see what changes are required, and it may help him/her to make a decision as to whether or not to use a particular component.

Table 10 Sub categories of 'Desirable characteristics of OSS'

Cat-6	Desirable characteristics of OSS	
Sub Category ID	Sub Category Name	Representative Quote
SC-6-1	Academic Perspective	"In academics because you are not delivering the PL for business purposes and the value is or the basis is extending the body of knowledge and helping other researchers to develop or break through in new area of software capabilities or demonstrating new algorithms or infrastructures whatever its purely functionality and functionality could be the number of test cases delivered with the open source product and the ease of use."
SC-6-2	Industrial Perspective	"…the critical business importance is not to take risks this is known as the risk assessment, how risky it is and for risk there are several ways of determining risks."
SC-6-3	Maintenance Support	"…support for the open source software is the number one characteristic we look for"
SC-6-4	Maintenance Agreement	"companies may opt for maintenance agreement with the developer company, at any time during the agreement if there is a need of extensions or change company can provide support"
SC-6-5	Infrastructure Support	"I would be looking for the software that has the capabilities that I want to include in my product line but also the capabilities that I need to support the services or the infrastructure"
SC-6-6	Maturity of OSS	"…we look for is how mature it is and if we have to change anything to make it work, how many examples of it are being used in software community"
SC-6-7	Error Handling Mechanism	"When an error happens, what type of interrupt /what type of message is passed back? What type of parameters is required to handle the error?"
SC-6-8	Scalability	"The biggest variability parameter, I am concerned with is the scale of the work… in other words, will this component scale?"

Table 11 Suggestions and representative quotes

Cat-7	Suggestions	
Sub Category ID	Sub Category Name	Representative quote
SC-7-1	Inter Language Reuse	"…for example if I want to extract Python code, if I use Python code for example I am a C# programmer that use dot net technology so how can I use it ... for example I use Python for text processing, how can I use Python code? How can it be portable or how can it be easily used in Java or C?"
SC-7-2	Software Agents	"…for software development there is such type of agent, that they can easily see other things also, if they are using some other code, agent can suggest which kind of change is required and which variable constant you should change or which type of features/ classes."

4. DISCUSSIONS

The adaptation of the coding process, i.e. the use of a word cloud, is a methodological contribution of this study. It may be used in other studies where textual data is analyzed. In this research, a word cloud is used after open coding to aid the process by ensuring that none of the recurring words related to a concept are missed. However, this technique can be used to pilot the open coding process, especially when a large amount of textual data is processed.

A word cloud can be more useful in the analysis of transcripts from unstructured interviews. The adaptation of the coding process may be used in grounded theory studies.

5. VALIDITY OF RESULTS

For qualitative research there are different types of validity: descriptive, interpretive, concurrent and theoretical validity [35]. Descriptive validity is related to the reporting of events, behaviours, settings, people, places and time. This is not much of a concern in this study. Interpretive validity is more of a concern for our research. Whenever there was ambiguity, the transcriptions were reviewed by the researcher to ensure the interpretive validity of the results. Furthermore, the findings of the qualitative studies are provided to the respondents. This measure was taken to cater for respondents' possible apprehension of the results. The respondents verified the interpretations.

Theoretical validity of the results is maintained by comparing the findings of this research study with contemporary studies. It can safely be said that the findings presented in this study are in line with the available theory.

Concurrent validity of the results is exhibited by the fact that the qualitative data were collected using seven interviews and similar patterns and trends were identified from the collected data. Only findings are reported that are concurrent, i.e. extracted from multiple respondents. The findings presented under the category 'suggestions' are subject to concurrent validity.

One of the category i.e. factors affecting reusability confirms and enhances our other works [36] in this field regarding the reusability of aspect oriented components. A summary of findings of this research is presented in Figure 3.

6. CONCLUSION

OSS has opened up new opportunities for reuse-intensive software development. This exploratory study was conducted to report the state of the art in this field. In this study, an exploratory research method, the interview, was used to gather data. Interpretation of the data was heavily influenced by what the literature says. The findings of the study are structured into seven categories and their 39 dimensions. These categories and their dimensions provide an in-depth view from the perspective of the potential user, the software engineer. Having distinguished respondents, with knowledge spanning both software houses and academia, ensures the credibility of the research. Apart from these findings, the study also makes a methodological contribution (an adaptation to the coding process). In future, studies may be carried out to further explore the identified issues reported in this study.

REFERENCES

[1] Niemi, T., *et al.*, "Server-Based Computing Solution Based on Open Source Software," *Information Systems Management*, vol. 26, pp. 77-86, 2009.

[2] Stafford, J. 2006, *Time to plan your company's escape from Microsoft*. Available: http://searchenterpriselinux.techtarget.com/news/1163576/2006-Time-to-plan-your-companys-escape-from-Microsoft, 01 June

[3] Kenwood, C. A., "A Business Case Study of Open Source Software," The MITRE Corporation2001.

[4] Krishnamurthy, S., "A Managerial Overview of Open Source Software," *Business Horizons*, vol. September-October 2003, 2003.

[5] Linden, F. v. d., *et al.*, "Commodification of Industrial Software: A Case for Open Source," *IEEE Software*, vol. 26, pp. 77-83, 2009.

[6] Wheeler, D. A. 2005, *Why Open Source Software / Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers!* Available: http://www.dwheeler.com/oss_fs_why.html, 01 June

[7] Howe, C., "Open Source Cracks The Code," Forrester Research2000.

[8] Ågerfalk, P. J., *et al.*, "Assessing the Role of Open Source Software in the European Secondary Software Sector: A Voice from Industry," presented at the First International Conference on Open Source Systems, Genova, 2005.

[9] Hummel, O., *et al.*, "Code Conjurer: Pulling Reusable Software out of Thin Air," *IEEE Software*, vol. 25, pp. 45-52, 2008.

[10] Wasserman, A., "How the Internet transformed the software industry," *Journal of Internet Services and Applications*, vol. 2, pp. 11-22, 2011.

[11] Sommerville, I., *Software Engineering*, 8th ed.: Addison-Wesley, 2007.

[12] Ågerfalk, P., *et al.*, "Open Source in Software Product Line: An Inevitable Trajectory," in *10th International Software Product Line Conference (SPLC '06)*, 2006.

[13] Ahmed, F., *et al.*, "A Model of Open Source Software-Based Product Line Development," in *Computer Software and Applications, 2008. COMPSAC '08. 32nd Annual IEEE International*, 2008, pp. 1215 -1220.

[14] Stol, K. J. and Babar, M. A., "Challenges in using open source software in product development: a review of the literature," presented at the Proceedings of the 3rd International Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, Cape Town, South Africa, 2010.

[15] Saunders, M., *et al.*, *Research Methods for Business Studies* 5th ed.: Prentice Hall, 2009.

[16] Strauss, A. and Corbin, J., *Basics of Qualitative Research Techniques and Procedures for Developing Grounded Theory*, 2nd edition ed.: Sage Publications, 1998.

[17] Seaman, C. B., "Qualitative Methods in Empirical Studies of Software Engineering," *IEEE Transactions on Software Engineering*, vol. 25, pp. 557-572, 1999.

[18] Gray, D. E., *Doing Research in the Real World*, 2nd ed.: SAGE Publication Ltd., 2009.

[19] Punch, K. F., *Introduction to Research Methods in Education* Sage Publications Ltd., 2009.

[20] atlas.ti. 2011, *atlas.ti*. Available: www.atlasti.com, 06, June

[21] Bongshin, L., *et al.*, "SparkClouds: Visualizing Trends in Tag Clouds," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, pp. 1182-1189, 2010.

[22] Höst, M., *et al.*, "Usage of Open Source in Commercial Software Product Development – Findings from a Focus Group Meeting," in *Product-Focused Software Process Improvement*. vol. 6759, D. Caivano, *et al.*, Eds., ed: Springer Berlin / Heidelberg, 2011, pp. 143-155.

[23] Schryen, G., "Is open source security a myth?," *Communications of the ACM*, vol. 54, pp. 130-140, 2011.

[24] Stol, K. J., *et al.*, "A comparative study of challenges in integrating Open Source Software and Inner Source Software," *Information and Software Technology*, vol. In Press, Corrected Proof, 2011.

[25] Haefliger, S., *et al.*, "Code Reuse in Open Source Software," *MANAGEMENT SCIENCE*, vol. 54, pp. 180-193, January 1, 2008 2008.

[26] von Krogh, G., *et al.*, "Knowledge Reuse in Open Source Software: An Exploratory Study of 15 Open Source Projects," in *System Sciences, 2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on*, 2005, pp. 198b-198b.

[27] Joode, R. v. W. d., *et al.*, "Rethinking free, libre and open source software.," *Knowledge, Technology & Policy*, vol. 18, pp. 5-16, 2006.

[28] Forge, S., "The rain forest and the rock garden: the economic impacts of open source software," *info*, vol. 8, pp. 12-31, 2006.

[29] Varian, H. R. and Shapiro, C., "Linux adoption in the public sector: An economic analysis (Technical Report)," UC Berkeley2003.

[30] Sojer, M. and Henkel, J., "Code Reuse in Open Source Software Development: Quantitative Evidence, Drivers, and Impediments," *Journal of the Association for Information Systems*, vol. 11, pp. 868-901, 2010.

[31] Lazaro, M. and Marcos, E., "An Approach to the Integration of Qualitative and Quantitative Research Methods in Software Engineering Research," in *2nd International Workshop on Philosophical Foundations of Information Systems Engineering (PHISE'06)*, 2006.

[32] Spinellis, D., "Choosing and Using Open Source Components," *IEEE Software*, vol. 28, pp. 96-96, 2011.

[33] Fazal-e-Amin, *et al.*, "An analysis of object oriented variability implementation mechanisms," *SIGSOFT Softw. Eng. Notes*, vol. 36, pp. 1-4, 2011.

[34] Fazal-e-Amin, *et al.*, "Metrics Based Variability Assessment of Code Assets " in *Software Engineering and Computer Systems*. vol. 181, J. M. Zain, *et al.*, Eds., ed: Springer Berlin Heidelberg, 2011, pp. 66-75.

[35] Johnson, B. and Christensen, L., *Educational Resaerch : Quantitative, Qualitative, and Mixed Approaches* 4th ed.: Sage Publication, Inc., 2011.

[36] Fazal-e-Amin, *et al.*, "A proposed reusability attribute model for aspect oriented software product line components," in *Information Technology (ITSim), 2010 International Symposium in*, 2010, pp. 1138-1141.