# Trends in Applied Sciences Research

ISSN 1819-3579

Academic
Journals Inc.

www.academicjournals.com

Trends in Applied Sciences Research 7 (2): 118-131, 2012
ISSN 1819-3579 / DOI: 10.3923/tasr.2012.118.131


# Relative Importance of Factors Constituting Component Reusability

Fazal-e-Amin, Ahmad Kamil Mahmood and Alan Oxley
Department of Computer and Information Sciences, University Technology Petronas, Bandar Seri Iskandar, 31750 Tronoh, Perak, Malaysia

*Corresponding Author: Fazal-e-Amin, Department of Computer and Information Sciences, University Technology Petronas, Bandar Seri Iskandar, 31750 Tronoh, Perak, Malaysia*

## ABSTRACT

The potential benefits of software reuse include a reduction in development time, effort, cost and an increase in quality and productivity. Reuse is carried out either by using component based software development or by systematic reuse as is employed by software product lines. Improved accessibility of open source software components has opened up new avenues for combining software components with software product lines. As a result, proposals for open source component based software product lines appear in the literature. This study is a step forward in this direction. An assessment approach for software component reusability is employed and reusability of several components is assessed. The results include a statistical analysis to determine the correlation between the factors of reusability. The results show a strong correlation of four out of six identified factors with reusability.

**Key words:** Software product line, software components, reusability assessment, open source, mixed-method, interview

## INTRODUCTION

Software reuse refers to the process of developing software by making use of existing software (Krueger, 1992). Reuse in software development is motivated by the factors of time, cost and effort. Software reuse results in better quality and productivity (Mohagheghi and Conradi, 2007). It reduces the cost and effort to develop software (Frakes and Succi, 2001). The two common forms in which reuse is employed are component based software development and Software Product Line (SPL) development. The latter is a systematic way of reuse. An SPL is defined as "development for the reuse and development with reuse" (Van der Linden *et al*., 2007). It is a reuse-intense software development where reuse is the main tenet.

SPLs, being a systemic way of software reuse, distinguish the common features of products from the unique ones. These commonalities are catered for by common components. The collection of common components is referred as core components. A key feature of such components is 'variability', i.e., the "degree to which something exists in multiple variants, each having the appropriate capabilities" (Firesmith, 2003).

Open Source Software (OSS) is one of the emerging areas within software engineering. It influences the way that software is developed (Hauge *et al*., 2010), not only at the development level but also on the procedural level. Component based software engineering is one of the main beneficiaries of OSS components. Recent research work (Ahmed *et al*., 2008) suggest the use of OSS in product line development. One of the reasons the SPL community is attracted towards OSS is

the fact that product lines are seldom started from scratch, rather, SPLs emerge when the domain is mature (Knodel *et al*., 2005). Therefore, OSS may provide an initial support to start new product lines. A SPL is reuse intense development; the focus is on reusing the core assets. The term 'core asset' refers to test cases, components, architecture or any common artefact shared by two or more products.

The importance of the Off The Shelf components (OTS) is reflected in a claim by Driver (2008) that "It is becoming not only impractical, but also virtually impossible for mainstream IT organizations to ignore the growing presence of third party software in major segments of the IT industry. The failure to optimally manage the potential risks and rewards of using this software will put IT organizations at an increasingly serious risk in coming years".

The benefits of using OSS are explored (Morgan and Finnegan, 2007). These include reliability, security, quality, performance, flexibility of use, flexibility in other respects, low cost, having a large developer and tester base, having community support in the form of users, increased collaboration and escaping vendor lock in, encouraging innovation. These benefits are contributing to the popularity of OSS.

Each artefact developed during SPL development is considered as an asset, the organization keeps this asset for reusing it in further products or even in cross product line reuse (Liu *et al*., 2010). OSS and OTS comes from numerous sources, therefore, the assessment of reusability has more importance in reuse intense developments. Further more, the success of a product line depends on the efficient reuse of core assets. This reuse of a core asset, i.e., its reusability, is related to the quality of the component. Reusability assessment also helps in comparing different components providing same functionality.

A review of the software component reusability assessment approaches is presented in (Fazal-e-Amin *et al*., 2011d), it categorizes approaches according to their types, approach, application level and validation. In this paper only the metrics based approaches are considered. A model and metrics for object oriented C$^{++}$ based implementations are presented (Etzkorn *et al*., 2001). In this paper three views of reusability are defined which are: reusability in class, reusability in a hierarchy/subsystem and reusability in the original system. Factors, sub factors and metrics are proposed to measure reusability. Results are validated using expert opinions about the reusability of the components, which is compared with results generated through the application of the proposed approach; regression analysis is used to interpret the results.

In (Cho *et al*., 2001) metrics to measure complexity, customizability and reusability of Java components are proposed. The degree of features reused in developing an application is used to measure reusability. Two types of metrics are proposed, one is the metrics to be used at the design phase and the second is the metrics used after coding-the number of lines of code; the proportion of overall functionality that each component has. These metrics are implemented on components of the banking domain. However, no validation of results is presented in the paper.

A model and set of metrics to measure reusability is presented (Dandashi, 2002); it considers adaptability, completeness, maintainability and understand ability as factors affecting reusability. These factors are measured by the metrics. These metrics are applied to the (C$^{++}$ based) components of a scientific application in order to evaluate the approach. The approach is validated by finding a correlation coefficient between the results of direct measure, using the results of the proposed approach and measures collected via the survey instrument.

Two metrics are proposed (Aggarwal *et al*., 2005) to measure the amount of generic code. The proposed metrics are applied to ten projects. The metrics are evaluated using Weyuker's properties.

Gui and Scott (2007) proposed coupling metrics to rank the reusability of components. Metrics are applied to three types of component to generate the results. The metrics to measure coupling (Gui and Scott, 2007) and cohesion (Gui, 2006) are combined by Gui and Scott (2009) to measure the reusability of software components. In (Gui and Scott, 2008), coupling and cohesion metrics are also proposed to evaluate the reusability of components. The statistical techniques of linear regression and rank correlation are used for validation of the results.

This study presents a correlation analysis of attributes and reusability. These attributes are identified by a literature survey and an exploratory study which is part of our future publications. In this study we are only considering the non-functional attributes.

## MATERIALS AND METHODS

The methodology comprises the following steps:

**Step 1:** An exploratory study is conducted which identifies the factors affecting reusability along with other findings

**Step 2:** The metrics were identified to measure the factors

**Step 3:** The hypotheses are generated on the basis of the literature and our findings during an exploratory study about the factors affecting the reusability

**Step 4:** The components are downloaded from different sources including (Planet-Source-Code.com and Merobase.com). A total of 75 classes are drawn from 12 components. Here it is important to mention that the classes were in a hierarchy (i.e., are related to each other). A random search was made during the time period January 2011 to February 2011, using the following queries (library system, banking, accounts and user management). The retrieved components were related to the following projects (address book, airline reservation system, menu builder, car sales system, class browser, flight reservation system, user bank account management system)

**Step 5:** The proposed metrics are applied to the components to collect the values

**Step 6:** These metric values are interpreted using the equations

**Step 7:** The scatter plots are drawn and a Pearson correlation test is applied to determine the correlation

**Interview:** A summary of the related work is presented in the previous section, apart from it, a detailed review of approaches can be found (Fazal-e-Amin *et al*., 2011d). Reusability assessment is not viewed in the context of software product lines. Therefore, the nature of the study is exploratory. So, an exploratory research method is used and the 'interview' is employed as a data collection tool.

The interview is a means of collecting primary data; it is a conversation between two persons, one of which 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 (Gray, 2009). Interviews may help to gather impressions and opinions about something. 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. Interviews are best for exploring the perspective of informants (Gray, 2009). In the context of this study, the informants are those who have experience with open source and product lines and preferably have academic/research experience. The authors have contacted several people and managed to conduct interview sessions with five informants. A brief introduction of them is presented in Table 1.

Table 1: Information about the respondents

| Respondent ID | Experience | Experience type | Current affiliation |
|---|---|---|---|
| Rsp-A | 05 years | Academic, Industrial | Academia |
| Rsp-B | 10 years | Academic, Industrial | Industry |
| Rsp-C | 22 years | Academic, Industrial | Industry |
| Rsp-D | 08 years | Academic, Industrial | Academia |
| Rsp-E | 10 years | Academic, Industrial | Academia |

The results are obtained using the content analysis approach (Hsieh and Shannon, 2005; Elo and Kyngas, 2008). Open coding is performed to get meaningful results. The results are divided into different categories. The details of the study cannot be presented here due to space limitations. However, the results relevant to this paper are presented here. The details can be found (Fazal-e-Amin *et al*., 2011e). The category that this paper is concerned with is factors affecting reusability of OSS in an SPL environment.

The scatter diagram gives a notion about the association. However, the correlation coefficient, or Pearson product-moment correlation coefficient, is a mathematical measure. It helps in understanding the strength of the relationship between the variables.

**Population of interview:** 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 population chosen for this study is 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. Table 1 provides a glance of the profiles of the respondents.

The means used to conduct the interviews are face to face (3), using Skype (1) and telephone (1). Three of the interviews were face to face, one was telephonic and one was conducted using Skype (online communication software). It was not possible to conduct all interviews face to face due to the geographical locations of the respondents.

Flexibility is related to reusability in two capacities. 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.

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.

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.

Another characteristic of the open source components explored is that the developer looks for a component covering more of the scope of the application. In some situations even the size does not matter but size is a concern in large sized components as it relates to increased complexity and poor understandability. Further more, scope coverage is important in situations where future enhancements are already envisioned or there are chances that more features would be added in future.

The interviewees consider stability as an important factor to be considered while making decisions. Here, the term 'stability' refers to security in numbers, that is, a reasonable number of

Table 2: Identified factors and representative quotes

| Factors affecting reusability | | |
|---|---|---|
| Sub category ID | Sub category Name | Representative quote |
| SC-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-2 | Maintainability | "Maintainability is a large problem is 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 others code that is developed by some other is very difficult for developer". |
| SC-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 softwares to run a component in other systems". |
| SC-4 | Scope Coverage | "That depends on the situation but normally we choose the more coverage component as compare 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 | Stability | "Stable meaning reasonably error free and it could be used with confidence that there is no bug". |
| SC-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-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-8 | Variability | "Variability is a two edge sword in other words there are advantages and disadvantages". |
| SC-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". |

developers have contributed in the development of the component and also it has been used by a reasonable number of developers. Stability is also related to the usage history of the component. Usage history provides a hint about the usefulness of the component. Another side of usage history is the maturity of the component.

The subjects also have a consensus on 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.

Variability is one of the factors; it decreases understandability. Variability is also seen as the configurability of a component, that it can be configured in multiple configurations.

The details of the exploratory study will be found in future publications of the authors, it is work in progress (Table 2).

**Sub-Attributes and metrics:** In this section a description of the attributes and metrics which are used to assess reusability is provided. The attributes, sub-attributes and their corresponding metrics are presented in Table 3.

Table 3: Attribute, sub-attributes and metrics

| Attribute | Sub-attribute | Metrics |
|---|---|---|
| Flexibility | Coupling, Cohesion | CBO, LCOM |
| Understandability | Coupling, Cohesion, Size | CBO, LCOM, %comments, LOC, NOM |
| Portability | Independence | DIT |
| Scope Coverage | | NOM ÷ Total number of methods |
| Maintainability | Complexity | MCC, MI |
| Variability | | NOC ÷ Total number of classes, NOM ÷ Total number of methods |

The factors and attributes are related to each other according to the following equations:

$$\text{Reusability of Class} = 0.16\times\text{Flexibility}+0.16\times\text{Understandability}+0.16\times\text{Portability}+0.16\times\text{Scope coverage}+0.16\times\text{Maintainability}+0.16\times\text{Variability}$$

$$\text{Flexibility} = 1-[(0.5\times\text{Coupling})+(0.5\times\text{Cohesion})]$$

$$\text{Coupling} = \text{adjusted CBO},\ \text{Cohesion} = \text{adjusted LCOM}$$

$$\text{Understandability} = 1-[(0.25\times\text{Coupling})+(0.25\times\text{Cohesion})+(0.25\times\text{Comments})+(0.25\times\text{Size})]$$

$$\text{Size} = (0.5\times\text{adjusted LOC})+(0.5\times\text{adjusted NOM})$$

$$\text{Portability} = \text{Independence} = 1-\text{adjusted DIT}$$

$$\text{Scope coverage} = \text{NOM} \div \text{Total number of methods in all classes}$$

$$\text{Maintainability} = (0.5\times\text{adjusted MCC})+(0.5\times\text{adjusted MI})$$

$$\text{Variability} = 0.5\times(\text{NOC} \div \text{Total number of classes})+0.5\times(\text{NOM} \div \text{Total number of methods in all classes})$$

As a starting point, equal weights/coefficients are assigned to each of the attributes and factors. Equal weights are used (Etzkorn *et al*., 2001) and it is stated that the linear combination of equal weights works well in most cases. Another example (Bansia and Devis, 2002) where equal weights are assigned to the attributes; this study was in the context of design quality.

**Maintainability:** In (IEEE, 2010) maintainability is defined as "the ease with which a software system or component can be modified to change or add capabilities, correct faults or defects, improve performance or other attributes, or adapt to a changed environment". Two metrics, MCC and MI, are used to measure maintainability.

**Portability:** It is defined as "the ease with which a system or component can be transferred from one hardware or software environment to another". The portability of a component depends on its independence, i.e., the ability of the component to perform its functionality without external support. In a scenario where an open source component is used in SPL development, the component should have the characteristic of portability. The component, being a core asset, may be used in the development of another product/family member within the product line/family.

**Flexibility:** It is defined as "the ease with which a system or component can be modified for use in applications or environments other than those for which it was specifically designed" (IEEE, 2010). In (Sant'Anna *et al*., 2003; Pohl *et al*., 2005; Sharma *et al*., 2009) flexibility is considered as a factor affecting the reusability of a component. In the context of an SPL, the flexibility characteristic is necessary for a core asset as it is intended to be reused in the development of other products.

**Understandability:** It is defined as "the ease with which a system can be comprehended at both the system-organizational and detailed statement levels" (IEEE, 2010). In (Sant'Anna *et al*., 2003; Washizaki *et al*., 2003) understandability is considered a factor of reusability.

**Scope coverage:** It is the attribute that measures the number of features provided by the component against the total number of features in the SPL scope.

**Independence:** The term 'independence' is introduced to reflect the property of the system concerning the ability of a class to perform its responsibilities on its own. Independence is measured by DIT. The classes lower in the hierarchy are inherited by other classes; these classes depend on their ancestors to perform their functionalities.

**Size metrics:** In (Fenton and Pfleeger, 1997) the aspect of the software dealing with its physical size is named the 'length' of the software. The metric used for size is Lines Of Code (LOC). It counts the lines of source code. The second metric used to measure size is Number Of Methods (NOM).

**Coupling and cohesion metrics:** Coupling and cohesion are two key concepts in object oriented software engineering. Both of these are related to interaction between the entities. The higher the level of interaction, the higher is the level of dependency. The lower the level of interaction, the higher is the level of cohesion. Cohesion refers to the extent to which an entity can perform its responsibilities on its own. The metric used for coupling is CBO and the one used for cohesion is LCOM.

**Variability metrics:** The mechanisms to introduce variability in object oriented systems are presented in (Fazal-e-Amin *et al*., 2011b) and an analysis of these mechanisms is presented in (Fazal-e-Amin *et al*., 2011a). On the basis of the analysis the variability metrics are presented and validated in (Fazal-e-Amin *et al*., 2011c). These metrics are used in this paper.

## RESULTS

This section includes the hypothesis, scatter plots and values of the correlation coefficient r. Table 4 contain the complete correlation statistics.

**Statistical test:** The scatter diagram gives a notion about the association. However, the correlation coefficient or Pearson product-moment correlation coefficient is a mathematical measure. It helps to understand the strength of the relationship between the variables.

The correlation coefficient r is a numerical measure that assesses the strength of the linear relationship between two variables. The following are assumptions of Pearson correlation coefficient r:

Table 4: Correlation between reusability and factors

| Correlations | Flexibility | Understandability | ScopeCov | Variability | Maintainability | Portability | Reusability |
|---|---|---|---|---|---|---|---|
| **Flexibility** | | | | | | | |
| Pearson Correlation | 1 | | | | | | |
| Sig. (2-tailed) | | | | | | | |
| N | 75 | | | | | | |
| **Understandability** | | | | | | | |
| Pearson correlation | 0.833** | 1 | | | | | |
| Sig. (2-tailed) | 0.000 | | | | | | |
| N | 75 | 75 | | | | | |
| **ScopeCov** | | | | | | | |
| Pearson correlation | -0.436** | -0.559** | 1 | | | | |
| Sig. (2-tailed) | 0.000 | 0.000 | | | | | |
| N | 75 | 75 | 75 | | | | |
| **Variability** | | | | | | | |
| Pearson correlation | -0.424** | -0.552** | 0.983** | 1 | | | |
| Sig. (2-tailed) | 0.000 | 0.000 | 0.000 | | | | |
| N | 75 | 75 | 75 | 75 | | | |
| **Maintainability** | | | | | | | |
| Pearson correlation | 0.494** | 0.558** | -0.318** | -0.293* | 1 | | |
| Sig. (2-tailed) | 0.000 | 0.000 | 0.006 | 0.011 | | | |
| N | 75 | 75 | 75 | 75 | 75 | | |
| **Portability** | | | | | | | |
| Pearson correlation | 0.213 | 0.264* | 0.043 | 0.072 | 0.340** | 1 | |
| Sig. (2-tailed) | 0.066 | 0.022 | 0.716 | 0.537 | 0.003 | | |
| N | 75 | 75 | 75 | 75 | 75 | 75 | |
| **Reusability** | | | | | | | |
| Pearson correlation | 0.744** | 0.711** | 0-.018 | -0.005 | 0.794** | 0.408** | 1 |
| Sig. (2-tailed) | 0.000 | 0.000 | 0.880 | 0.966 | 0.000 | 0.000 | |
| N | 75 | 75 | 75 | 75 | 75 | 75 | 75 |

**Correlation is significant at the 0.01 level (2-tailed). *Correlation is significant at the 0.05 level (2-tailed)

- r ranges from +1 to -1 i.e., $-1 \leq r \leq +1$. The value 1 shows a perfect positive linear correlation, while the value -1 shows a perfect negative linear correlation. The value 0 represents an absence of any linear correlation
- A positive value of r is an indication that y will increase with the increase in x. On the other hand, a negative value of r implies that the value of y will decrease when the value of x increases
- r is not affected by the order of x and y, i.e. r is the same for the pairs (x, y) and (y, x)
- r is not affected by a change in the units of the variables

The correlation coefficient r is measure the strength of the association between two variables. However, it does not implicate about the cause and effect. In other words the two variables x, y having a strong correlation and increasing or decreasing together does not mean that x is cause of increase/increase in y.

**p-value:** The p-value (probability value) represents the statistical significance of the test. The smaller the value of p the smaller is the likelihood that the null hypothesis holds. The p-value is

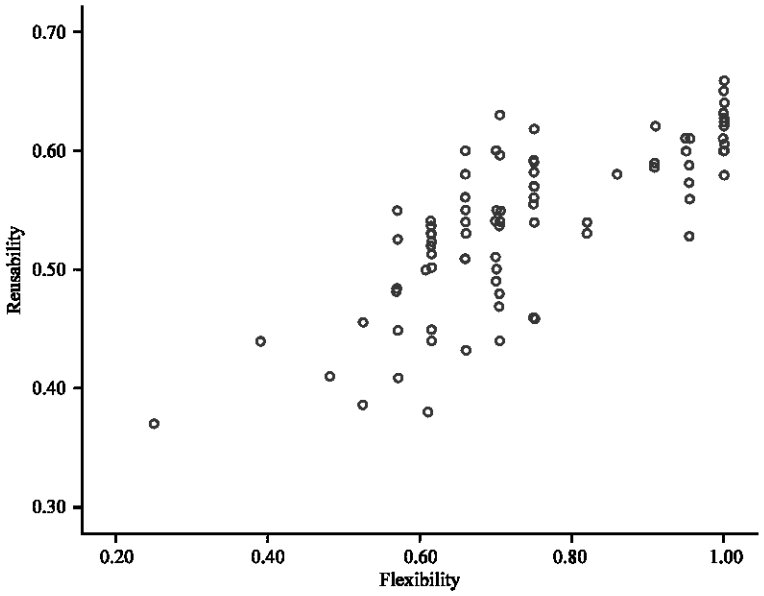


Fig. 1: Scatter plot of flexibility/reusability

compared against the alpha values. The commonly used alpha values are 0.05 and 0.01. A p-value less than 0.01 shows that the probability of the null hypothesis being true is 1 time in 100 samples. A p-value less than 0.05 implies that the probability of the null hypothesis being true is 5 times in 100 samples. These values of alpha (0.01, 0.05) are lower levels of the alpha value. The null hypothesis can be rejected when the p-value falls below the chosen level. The p-values are mentioned in Table 4 in the row with the title 'sig'.

**Hypothesis tests:**

- **$H0_1$:** Flexibility of software has no effect on its reusability
- **$H1_1$:** Flexibility of software has an effect on its reusability

The correlation between flexibility and reusability is r (75) = 0.744, p = 0. It shows a strong positive correlation between flexibility and reusability. So, the null hypothesis is rejected and it can be concluded that flexibility is positively correlated to reusability. An increase in the value of flexibility increases reusability (Fig. 1).

- **$H0_2$:** Variability of software has no effect on its reusability
- **$H1_2$:** Variability of software has an effect on its reusability

The correlation between variability and reusability is r (75) = -0.005, p = 0.966. There is a weak negative correlation between variability and reusability; further, p>0.05 shows how insignificant the link is between variability and reusability. The correlation analysis leads to the rejection of the alternate hypothesis. It is concluded that variability is not related to reusability in this context. This conclusion demands further validation which may mean going back and rethinking about the variability metrics (Fig. 2):

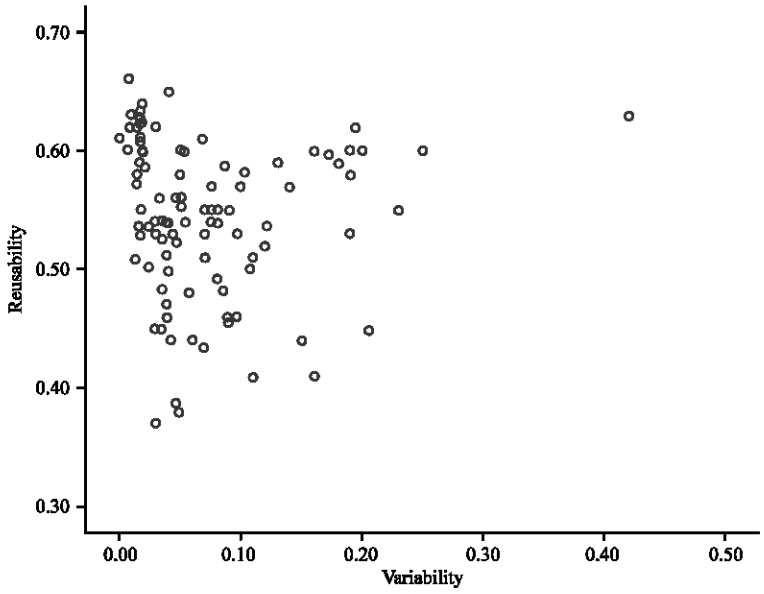


Fig. 2: Scatter plot of variability/reusability

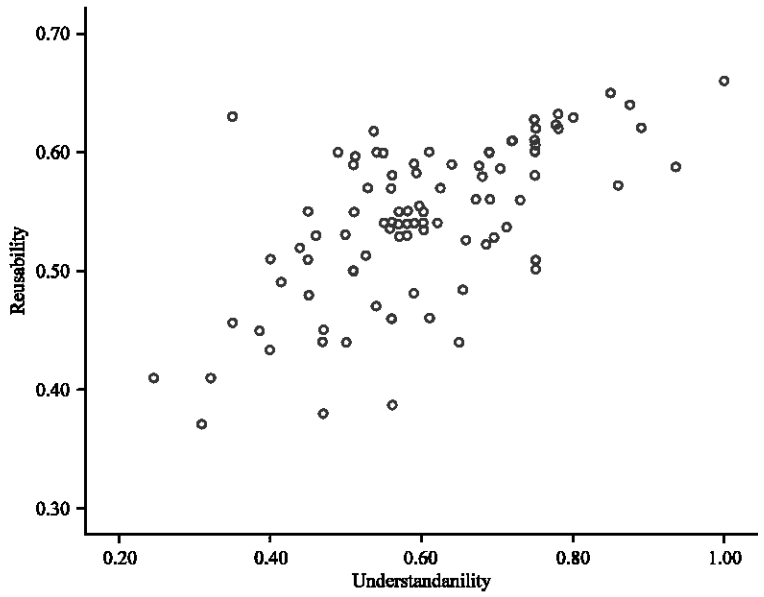


Fig. 3: Scatter plot of understandability/reusability

- **$H0_3$:** Understandability of software has no effect on its reusability
- **$H1_3$:** Understandability of software has an effect on its reusability

The correlation between understandability and reusability is r (75) = 0.711, p = 0. The value of r shows a strong positive correlation between understand ability and reusability. We reject the null hypothesis and it can be concluded that an increase in the value of understandability increases reusability (Fig. 3).

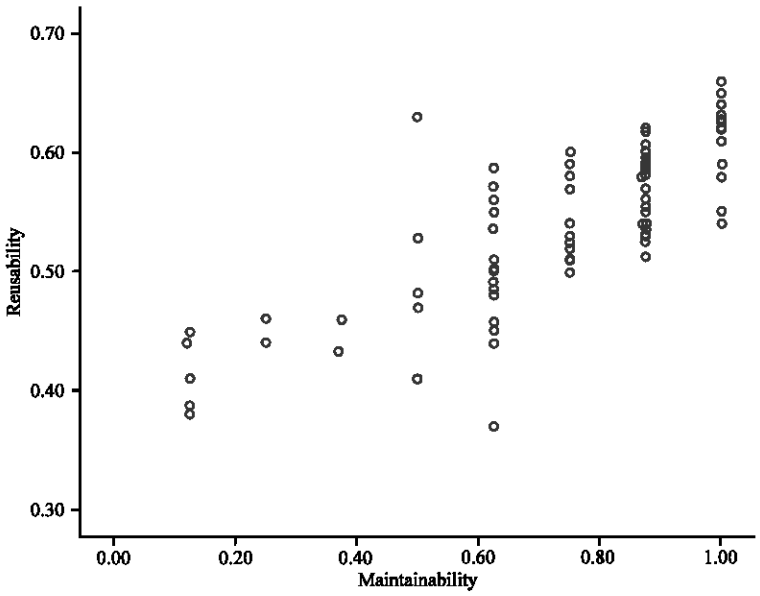


Fig. 4: Scatter plot of maintainability/reusability

- **$H0_4$:** Maintainability of software has no effect on its reusability
- **$H1_4$:** Maintainability of software has an effect on its reusability

The correlation between maintainability and reusability is r (75) = 0.794, p = 0. The r value shows a strong positive correlation between maintainability and reusability. The null hypothesis is rejected and it can be concluded that an increase in maintainability increases reusability (Fig. 4).

- **$H0_5$:** Portability of software has no effect on its reusability
- **$H1_5$:** Portability of software has an effect on its reusability

The correlation between portability and reusability is r (75) = 0.408, p = 0. The r value shows a weak positive correlation between portability and reusability. The value of p is 0, which leads to the rejection of null hypothesis. It can be concluded that there is a positive effect of portability on reusability. Increasing value of portability increases reusability (Fig. 5).

- **$H0_6$:** Scope-coverage of software has no effect on its reusability
- **$H1_6$:** Scope-coverage of software has an effect on its reusability

The correlation between scope coverage and reusability is r (75) = -0.018, p = 0.88. The r value shows a weak negative correlation between scope coverage and reusability. However, the inequality $p < 0.05$ shows how insignificant this relationship is. Therefore, the alternate hypothesis is rejected and it can be concluded that scope coverage is not related to reusability in this context. These results demand further investigation (Fig. 6).

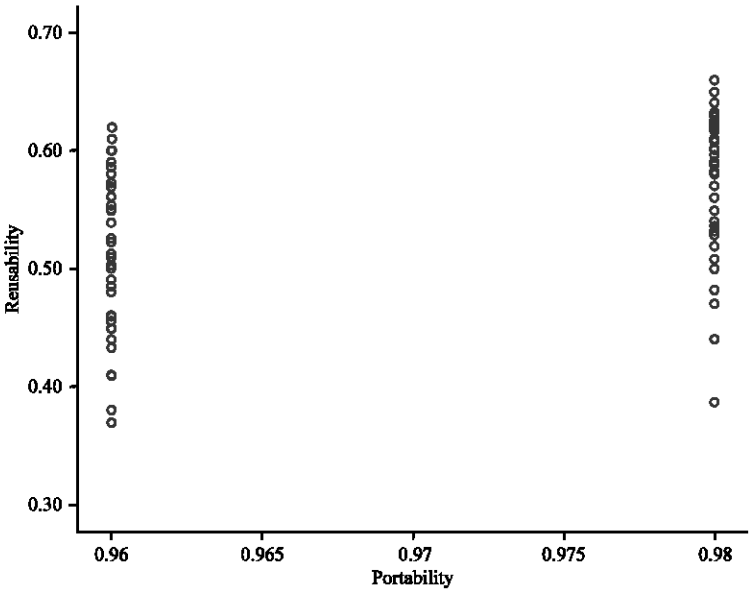


Fig. 5: Scatter plot of portability/reusability

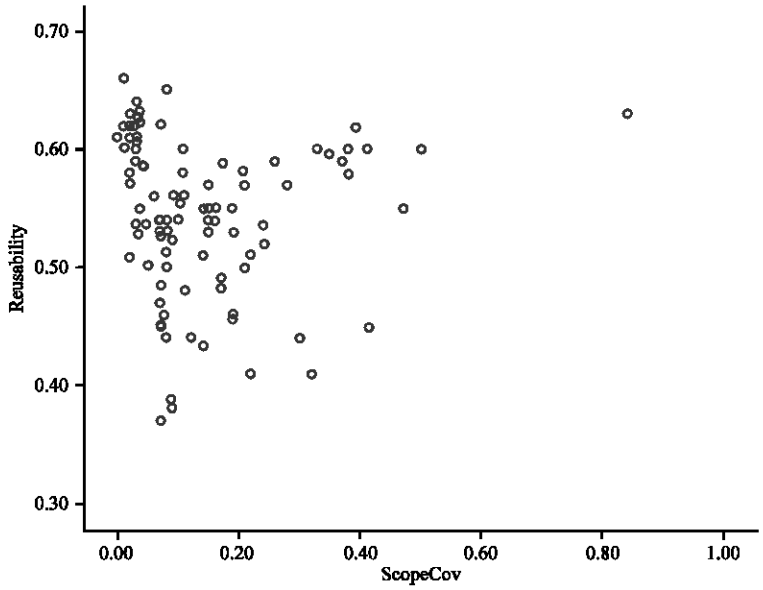


Fig. 6: Scatter plot of scope coverage/reusability

## CONCLUSIONS

The growing interest of research into open source components based SPLs and in reuse intense software development is reflected by the number or recent papers appearing in the literature. The emergence of open source as a contender for industrial software development attracts the SPL community. In this paper, the factors of reusability are measured using established object oriented metrics. A statistical analysis is performed to gain further insight. On the basis of the results, it is concluded that, in our context, flexibility, understand ability, maintainability and portability are

positively correlated to reusability, while scope coverage and variability are not reusability. These initial results may provide a foundation for further exploration in this area. Future work will involve finding more metrics to measure variability and scope coverage and reassessing their relationships with reusability.

## REFERENCES

Aggarwal, K.K., Y. Singh, A. Kaur and R. Malhotra, 2005. Software reuse metrics for object-oriented systems. Proceedings of the 3rd ACIS International Conference on Software Engineering Research, Management and Applications, August 11-13, 2005, IEEE Computer Society, Washington, DC, USA., pp: 48-55.

Ahmed, F., L.F. Capretz and M.A. Babar, 2008. A model of open source software-based product line development. Proceedings of the 32nd Annual IEEE International Computer Software and Applications, July 28-August 1, 2008, Turku, pp: 1215-1220.

Bansia, J. and C.G. Devis, 2002. A hierarchical model for object-oriented code in design quality assessment. IEEE Trans. Software Eng., 28: 4-17.

Cho, E.S., M.S. Kim and S.D. Kim, 2001. Component metrics to measure component quality. Software engineering conference. Proceedings of the 8th Asia-Pacific on Software Engineering Conference December 4-7, 2001 IEEE Computer Society, Washington, DC, USA., pp: 419-426.

Dandashi, F., 2002. A method for assessing the reusability of object-oriented code using a validated set of automated measurements. Proceedings of the ACM Symposium on Applied Computing, Madrid, Spain, March 11-14, 2002 ACM, New York, USA., pp: 997-1003.

Driver, M., 2008. New research: Predicts 2009: The evolving open-source software model. Gartner, http://blogs.gartner.com/mark_driver/2008/12/08/new-research-predicts-2009-the-evolving-open-source-software-model/.

Elo, S. and H. Kyngas, 2008. The qualitative content analysis process. J. Adv. Nurs., 62: 107-115.

Etzkorn, L.H., W.E. Hughes and C.G. Davis, 2001. Automated reusability quality analysis of OO legacy software. Inform. Software Technol., 43: 295-308.

Fazal-e-Amin, A.K. Mahmood and A. Oxley, 2011a. An analysis of object oriented variability implementation mechanisms. ACM SIGSOFT Software Eng. Notes, 36: 1-4.

Fazal-e-Amin, A.K. Mahmood and A. Oxley, 2011b. Mechanisms for managing variability when implementing object oriented components. Proceedings of the National Information Technology Symposium (NITS), October 9-11, 2011, King Saud University.

Fazal-e-Amin, A.K. Mahmood and A. Oxley, 2011c. Metrics based variability assessment of code assets. Commun. Comput. Inform. Sci., 181: 66-75.

Fazal-e-Amin, A.K. Mahmood and A. Oxley, 2011d. A review of software component reusability assessment approaches. Res. J. Inform. Technol., 3: 1-11.

Fazal-e-Amin, A.K. Mahmood and A. Oxley, 2011e. Using open source components in software product lines: An exploratory study. Proceedings of the IEEE Conference on Open Systems, September 25-28, 2011, Langkawi, Malaysia.

Fenton, N.E. and S.L. Pfleeger, 1997. Software Metrics: A Rigorous and Practical Approach. 2nd Edn., Thomson Asia, Singapore, ISBN: 9789812403858, Pages: 638.

Firesmith, D.G., 2003. Common concepts underlying safety, security and survivability engineering. Software Engineering Institute, http://www.sei.cmu.edu/library/abstracts/reports/03tn033.cfm.

Frakes, W.B. and G. Succi, 2001. An industrial study of reuse, quality and productivity. J. Syst. Softw., 57: 99-106.

Gray, D.E., 2009. Doing Research in the Real World. 2nd Edn., SAGE Publication Ltd., London, ISBN: 9781847873378, Pages: 604.

Gui, G. and P.D. Scott, 2007. Ranking reusability of software components using coupling metrics. J. Syst. Software, 80: 1450-1459.

Gui, G. and P.D. Scott, 2008. New coupling and cohesion metrics for evaluation of software component reusability. Proceedings of the 9th International Conference for, Young Computer Scientists, November 18-21, 2008 IEEE Computer Society, Washington, DC, USA., pp: 1181-1186.

Gui, G. and P.D. Scott, 2009. Measuring software component reusability by coupling and cohesion metrics. J. Comput., 4: 797-805.

Gui, G., 2006. Component reusability and cohesion measures in object-oriented systems. Inform. Commun. Technol., 2: 2878-2882.

Hauge, O., C. Ayala and R. Conradi, 2010. Adoption of open source software in software-intensive organizations: A systematic literature review. Inform. Software Technol., 52: 1133-1154.

Hsieh, H.F. and S.E. Shannon, 2005. Three approaches to qualitative content analysis. Qual. Health Res., 15: 1277-1288.

IEEE, 2010. Systems and software engineering: Vocabulary. ISO/IEC/IEEE 24765:2010(E). Institute of Electrical and Electronics Engineers, USA.

Knodel, J., I. John, D. Ganesan, M. Pinzger, F. Usero, J.L. Arciniegas and C. Riva, 2005. Asset recovery and their incorporation into product lines. Proceedings of the 12th Working Conference on Reverse Engineering, November 7-11, 2005, Pittsburgh, PA, USA., pp: 10-10.

Krueger, C.W., 1992. Software reuse. ACM Comput. Surv., 24: 131-183.

Liu, Y., K. Nguyen, M. Witten and K. Reed, 2010. Cross product line reuse in component-based software engineering. Proceedings of the International Conference on Computer Application and System Modeling, October 22-24, 2010, Taiyuan, pp: V9-427-V9-434.

Mohagheghi, P. and R. Conradi, 2007. Quality, productivity and economic benefits of software reuse: A review of industrial studies. Empirical Software Eng., 12: 471-516.

Morgan, L. and P. Finnegan, 2007. Benefits and drawbacks of open source software: An exploratory study of secondary software firms. Open Source Dev. Adoption Innov., 234: 307-312.

Pohl, K., G. Bockle and F. van der Linden, 2005. Software Product Line Engineering: Foundations, Principles and Techniques. Birkhauser, Heidelberg, ISBN: 9783540243724, Pages: 467.

Sant'Anna, C., A. Garcia, C. Chavez, C. Lucena and A. von Staa, 2003. On the reuse and maintenance of aspect-oriented software: An assessment framework. Proceedings of the XVII Brazilian Symposium on Software Engineering, (BSSE'03), UFBA, Computer Science Department, pp: 19-34.

Sharma, A., P.S. Grover and R. Kumar, 2009. Reusability assessment for software components. ACM SIGSOFT Software Eng. Notes 34: 1-6.

Van der Linden, F., K. Schmid and E. Rommes, 2007. Software Product Lines in Action: The Best Industrial Practice in Product Line Engineering. Springer, New York, ISBN: 9783540714361, Pages: 333.

Washizaki, H., H. Yamamoto and Y. Fukazawa, 2003. A metrics suite for measuring reusability of software components. Proceedings of the 9th International Symposium on Software Metrics, September 3-5, 2003 Sydney, Australia, pp: 211-225.