State-of-the-Art Object-Oriented Metrics and Its Reusability: A Decade Review

Chapter · January 2018

DOI: 10.1007/978-981-10-5544-7_42

3 authors:

Dr. Neelamadhab Padhy
GIET University
136 PUBLICATIONS **1,412** CITATIONS
SEE PROFILE

Suresh Satapathy
KIIT University
48 PUBLICATIONS **1,163** CITATIONS
SEE PROFILE

R.P. Singh
Sri Satya Sai Institute of Science & Technology
75 PUBLICATIONS **541** CITATIONS
SEE PROFILE

State-of-the-Art Object-Oriented Metrics and Its Reusability: A Decade Review

Neelamadhab Padhy, Suresh Satapathy and R. P. Singh

Abstract In this article, the importance is given to the object-oriented metrics and its reusability factor. There has been much work already done, but some improvements are needed in the software industry. Object-oriented metrics is one of the most popular and ongoing studies in different engineering branches including mathematics. Software metrics will be helpful to estimate the reusable code. The objective of this paper was to identify the reusability factor and accessibility from the last decades. A comprehensive survey on metrics and its applications has been carried out for more than one decade, and the main aim of this survey is to point out the reusable factor rather than coding. This paper shows the competence and applicability in a mixture of domains.

Keywords Object-oriented metrics · Reusability · Metrics model

1 Introduction

The term metrics is all about the measurement in the software code, and nowadays, object-oriented metrics (OOM) plays a crucial role in the industry and it is being used as a research tool. Researchers use it as a well-defined positive method to examine the data organization from the database. Several properties of the metrics have been defined

N. Padhy (✉)
Computer Science and Engineering, Satya Sai University of Technical and Medical Science (SSSUTMS), Sehore, Madhya Pradesh, India
e-mail: neela.mbamtech@gmail.com

S. Satapathy
P.V.P.Siddhartha Institute of Engineering and Technology, Kanuru, Vijayawada, Andhra Pradesh, India
e-mail: sureshsatapathy@ieee.org

R. P. Singh
Satya Sai University of Technical and Medical Science (SSSUTMS), Sehore, Madhya Pradesh, India

S. C. Satapathy et al. (eds.), *Smart Computing and Informatics*, Smart Innovation, Systems and Technologies 77, https://doi.org/10.1007/978-981-10-5544-7_42

so far, but still this measurement is not sufficient. The researchers put more emphasis on its software quality assurance. Whatever the metrics are defined, it is practically used in the programming languages. Significant work has already been done in this field, but the problem lies in the quality. To achieve a quality product is too difficult in the software industry; to fill this gap, we must deliver the good product. This can be achieved if the reusable components are provided so that time, effort and staffing are minimized. In fact, it is really a big challenge in this twenty-first century. Quality development of the software is an ongoing process; it cannot be achieved overnight. There are really two challenges for the programmers in the industry: bugs identification and rectification. Even though automated software does all the things, but measuring the quality attributes is still a challenging task from the code repositories. To find the quality attribute from the software programs, Kemerer and Chidamber [1] have developed a metric called the CK metrics suite. In-depth research has been conducted so far about the measurement and quality product [2–5]. There are other sources available, such as books and the Internet, which describe the metrics measurement [6, 7]. A major attempt has been initiated to produce the eminence of the product. At the end of the software development life cycle (SDLC), quality products will be extensively accepted; quality of the product depends on proper testing of the code. Quality is directly proportional to testability which gives satisfaction to the customer. The role of reusability plays a crucial role in this era. It is not a new concept. It is widely used for estimation of the software assessment. If the component is not reusable, then the whole concept of SDLC will fail. The new product will be developed by the existing one. A survey has been conducted on reusability.

The prime objective of this literature survey is to represent the current state-of-the-art software reusability metrics. Different researchers' views about the reusability are different.

2 Research Structure

In this paper, we investigated and formulated the followings:

Research Goal 1: What are the most reusable assets rather than coding?
Research Goal 2: How to estimate reusability? What are the methods or approaches for reusability?
Research Goal 3: How to validate the reusable products? What are the steps required to validate the reusable products?

3 Workflow Model for Reusability Evaluation

This is the proposed model for designing and developing the object-oriented paradigm; besides, this model provides the quantification of the reusability factor in the source codes (Fig. 1).

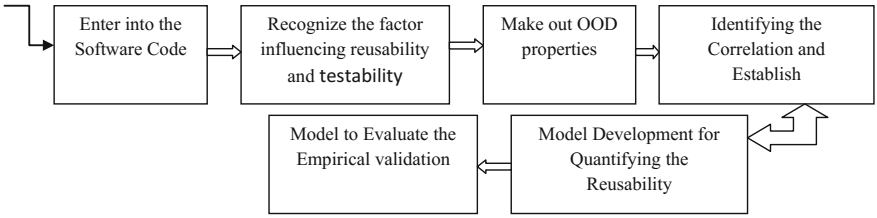
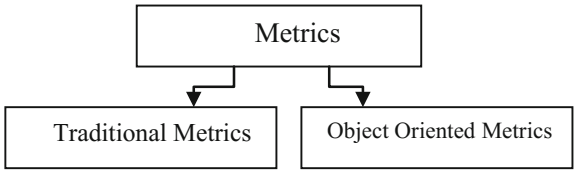


Fig. 1 Workflow model

Fig. 2 Metrics classification



Reusability occurs through the inheritance. The object-oriented design includes features such as inheritance, encapsulation, coupling, and cohesion. One of the important properties like encapsulation indicates hides the internal structure of the program. The above research framework demonstrates the quantification of the reusability. First of all, the challenging task is to identify the factors that influence reusability and testing and then to identify the properties of object-oriented design (OOD) metrics. OOD metrics help us to describe the quantification process of reusability as well as establish a multivariate linear model for reusability. The reusability factor can be achieved by using the other metrics: inheritance, cohesion, coupling, and encapsulation. This can be pictorially quantified and represented. During this study, we found different types of metrics. These are broadly classified into two types (Fig. 2).

4 Reusability Assets

In this paper, we only focused on the object-oriented metrics in terms of reusability and its assets. Software resources are one of the building blocks of the program paradigm; it can be one of them financial, economical. Not only the programs but also other things are reused. These are listed below. The reusable assets may consist of a single asset or several assets in one asset [8]. In this literature survey, we found 12 items that are reused apart from program code. These are as follows:

1. Used in the data
2. Modules in the program
3. Architecture-driven approach

4. Algorithms used in the program
5. Design patterns
6. Documentation for the project
7. Knowledge requirement
8. Models in the project
9. Planning stage
10. Requirement analysis
11. Service contracts

Table 1 Reusable properties

Reusability factor name	Meaning of the factor
1. Used in the data project (UD)	This indicates that the data can be reused frequently, thus achieving the target. Data mean an experience that is recorded during the previous projects [9]
2. Modules in the program (MIP)	A project is divided into several modules which contain the set of instructions. "Module" implies a single executable file that is only a part of the application, such as a DLL [10]
3. Architecture driven approach (ADP)	It is the approach which represents the overall structure of the project or a component
4. An algorithm used in the program (AP)	It is the reuse of the algorithms if the same type of problem occurs in the picture. Reusable algorithms are used in software designs [11]
5. Design patterns (DP)	The existing design will be reused if the same type of requirement occurs
6. Documentation in project (DIP)	It is one of the assets in the project. Documentation will be done during the SDLC, and a requirement specification analysis (RSA) will be done. New documents are designed which often share features of the old ones. All these are to reduce time and cost [12–14]
7. Knowledge requirement (KR)	During SDLC, knowledge will be generated and treated as one of the most prominent assets for the software component. The knowledge may represent the experience, idea, or reasoning [15–18]
8. Models in the project (MP)	A model can represent the task of the project, and it can consist of meaningful codes. It should also be able to represent the solutions and insights
9. Requirement analysis (RA)	It is the process of gathering the requirements for the projects. A requirement is a condition or capability that must be met or possessed by a system, product, service, result or component to satisfy a contract, standard, specification, or other formally imposed document [19]
10. Service contracts (SC)	It is the two-way communication between the two parties, i.e. the developers and users who are going to reuse the products. Hence, it is termed the reusable interface. This information can be helpful in predicting where and how the system can be tested, what problems might occur, and how to rectify the problem, after the system is evolved [20]
11. Test cases/test design (TCTD)	After the designing stage, the tester will develop the set of test cases, which is called as test case suite, and it can further be reused. They can be reused many times for different versions belonging to the same family [21–23]

12. Test cases/test design

From the above literature survey, we found the property used by the author in different aspects in different contexts (Table 1).

From the above studies, the data sets are created to recognize the most valuable assets for reusability. It is concluded that most of the researchers are using the requirement analysis, to which more attention has been paid in this twenty-first century.

5 Data Set for Reuse Assets

See Table 2.

6 Performance of Reusability Assets

See Fig. 3.

This graph is called the surface graph, and it represents the assets for the reusability during the last decades. Numerous researchers have studied in the different studies papers per year (SMPY). During the years 1991–1999, more researchers have shown their efforts (e.g., UDPO, MIP, DP, DIP, AP, KR, RA, and TCTD). Interestingly, almost all of them put forth their efforts in the RA, DIP, and ADP. Apart from these, during the period 2000–2009 the researchers extensively used other kinds of assets such as MIP and SC.

Table 2 Data set of reusable assets

S. No.	Name of the reusable assets	No. of articles
1	UP	05
2	MIP	04
3	ADP	11
4	AP	01
5	DP	10
6	DIP	09
7	KR	07
8	MP	01
9	RA	19
10	SC	02
11	TCTD	10

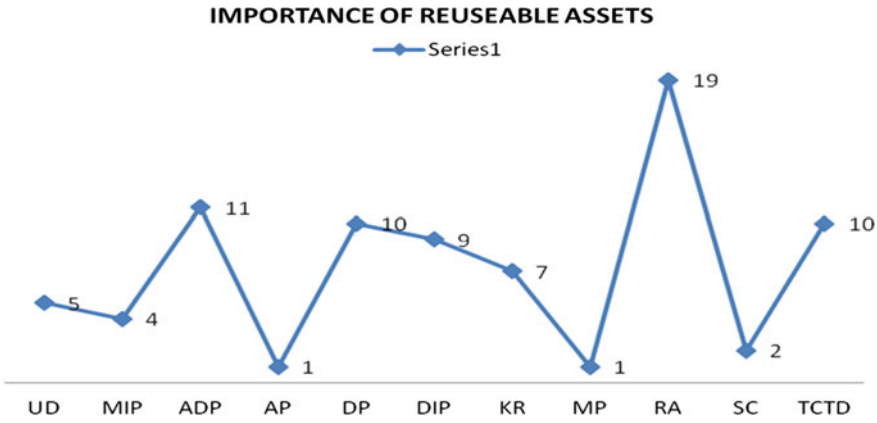


Fig. 3 Reusable assets

7 Conclusion and Future Scope

The prime objective of this literature review was to investigate the reusable assets. We have fully focused on the assets and found 11 most reusable attributes in this paper. During the investigation, we found that some of the articles are mentioned as nonvalidated, but they have not proved and claimed as reusable. Hence, extensive survey is required. Our future work is all about the extensive survey concerned with the industry-oriented reusable software analysis and maintenance, because our task is to reduce the maintenance cost and staffing save time. In the industry, about 65% of fund is invested for maintenance purposes. Further, this work is to enhance the understanding of how to maintain the reusable code in the industry. Further in-depth investigation is required in the manufacturing department in the industry where requirement and design are to be more emphasized. We found that most of the researchers observed that reusable assets are validated strongly in academics but poorly in the production sector. From this survey, researcher have shown that only 36% are validated and about 60% are nonvalidated, and the rest of the researchers have shown their kin interest to review studies. Again; comparison between academics and industry, then industry-oriented survey is less than academic. Not only reusability plays a vital role in the industry, but also the new kind of challenge is aging.

Appendix

Tables 3, 4, and 5 indicates that if studies then that article is represented as ($\surd$) mark otherwise it is indicated as (**x**). In this paper, we have investigated the limited attributes.

Table 3 Algorithms in the program (AP)

SL	Year	Category	Validation done or not					Not valid	Metrics	Module	Discussion		Review	Description about an algorithm used in this paper	Reference
			Industrial case study	Academic case study	Academic experience	Industry experience	Survey				Approach	Mention			
1	97	Algorithm	X	X	X	X	X	X	X	X		√		Represented an algorithm and stated that an algorithm can also be reused. Focused on the main target of algorithms	[11]

Table 4 Data used in the project (UD)

S. No.	Year	Category	Validation done or not					Not valid	Metrics	Module	Discussion		Review	Description about data used in this paper	References
			Industrial case study	Academic case study	Academic experience	Industry experience	Survey				Approach	Mention			
1	93	Used Data in the project	X	X	X	X	X	√	X	X	√	–	–	It is based on the data and focuses on how it is reused	[25]
2	94		X	X	X	X	X	X	X	X	X	–	√		[9]
3	96		X	X	X	X	X	X	X	X	X	–	√		[26]
4	97		X	X	X	X	X	X	X	X	X	–	√		[17]
05	04	Algorithm	X	X	X	X	X	X	X	X	X	X	√		[27]

Table 5 Models in the project—MP

S. No.	Year	Category	Validation done or not					Not valid	Metrics/tools	Module	Discussion		Review	Description about models used in this paper	Reference
											Approach	Mention			
		Models in the project	Industrial case studey	Academic case study	Academic experience	Industrial experience	Survey								
1	06		X	X	X	X	X	X	√	√	X	X	-	Refer the models and tools for reorganizing the reusable assets, representing and how to reuse the models	[24]

References

1. Kemerer, C.F., Chidamber, S.R.: A metrics suite for object oriented design. IEEE Trans. Softw. Eng. **20**(6), 476–493 (1994)
2. Henry, S., Li, W., Kafura, D., Schulman, R.: Measuring object-oriented design. J. Object Oriented Program **8**(4), 48–55 (1995)
3. Lorenz, M., Kidd, J.: Object-Oriented Software Metrics. Prentice Hall Object-Oriented Series. Prentice Hall, Englewood Cliffs, NJ (1994)
4. Henderson-Sellers, B.: Object-Oriented Metrics: Measures of Complexity. Prentice Hall, Englewood Cliffs, NJ (1996)
5. Singh, Y., Kaur, A., Aggarwal, K.K., Malhotra, A.: Empirical study of object-oriented metrics. J Object Technol. **5**(8), 149–173 (2006)
6. Fenton, N., Pfleeger, S.L.: Software Metrics: A Rigorous and Practical Approach, 2nd edn. International Thomson Computer Press, London (1997)
7. Shepperd, M.J., Ince, D.: Derivation and Validation of Software Metrics. Clarendon Press, Oxford (1993)
8. Hussein, K.: Measuring Reuse Characteristics of Software Components in an Extensible IDE, pp. 16–17. VDM Verlag, Saarbrücken (2008)
9. Visaggio, G.: Process improvement through data reuse. Softw. IEEE **11**(4), 76–85 (1994)
10. Capiluppi, A., Boldyreff, C.: Coupling patterns in the effective reuse of open source software. In: Proceedings of the First international Workshop on Emerging Trends in FLOSS Research and Development (20–26 May 2007). FLOSS. IEEE Computer Society, Washington, DC (2007)
11. Karsten, W.: Reuse of algorithms: still a challenge to object-oriented programming. In: Proceedings of the 12th ACM SIGPLAN Conference on Object Oriented Programming, Systems, Languages, and Applications. ACM, Atlanta, GA (1997)
12. Childs, B., Sametinger, J.: Literate programming and documentation reuse. In: Proceedings of the Fourth International Conference on Software Reuse, pp. 205–214 (1996)
13. Levy, D.M.: Document Reuse and Document Systems, vol. 6, no. 4, pp. 339–348. Electronic Publishing (1993)
14. Gil, J., Barta, D.: A system for document reuse. In: Proceedings of the 7th Israeli Conference on Computer systems and Software Engineering, pp. 83–94. IEEE Computer Society Press, Washington, DC (1996)
15. Arango, G., Schoen, E.: Design as evolution and reuse. In: Software Reusability, 1993. Proceedings Advances in Software Reuse (1993)
16. Hall, P.A.V.: Software components and reuse. Comput. Bull. **3**(4), 14–15 (1987)
17. Yglesias, K.P.: Information reuse parallels software reuse. IBM Syst. J. **32**(4), 615–620 (1993)
18. Soundarajan, S.F.N.: Inheritance: from code reuse to reasoning reuse. In: Fifth International Conference on Software Reuse ICSR'98, p. 206 (1998)
19. IEEE standard glossary of software engineering terminology. IEEE STD 610.12–1990, 1 (1990)
20. Lucas, C., Steyaert, P.: Managing software evolution through reuse contracts. In: Software Maintenance and Reengineering, 1997. EUROMICRO 97 (1997)
21. Mark, F., Lamey, T.: Common Test Patterns and Reuse Test Designs. Microsoft, Redmond (2008)
22. Mraz, T., Anneliese, V., Mayrhauser, R.: Domain based testing: increasing test case reuse. In: Proceedings of the IEEE International Conference on Computer Design, pp. 484–491. Cambridge, MA (1994)
23. Lonngren, D.D.: Reducing the cost of test through reuse. In: AUTOTESTCON'98. IEEE Systems Readiness Technology Conference, pp. 48–53. IEEE (1998)
24. Larsen, G.: Model-driven development: assets and reuse. IBM Syst. J. **45**(3), 541–553 (2006)

25. Jones, C.: Software Return on Investment Preliminary Analysis. Software Productivity Research Inc, Hendersonville (1993)
26. Frakes, W., Terry, C.: Software reuse: metrics and models. ACM Comput. Surv. **28**, 415–435 (1996)
27. Issenin, I., Brockmeyer, E., Miranda, M., Dutt, N.: Data reuse analysis technique for software-controlled memory hierarchies. In: Proceedings of the Conference on Design, Automation and Test in Europe, vol. 1, p. 10202. IEEE Computer Society, Washington, DC (2004)