J. Software Engineering & Applications, 2010, 3, 536-540
doi:10.4236/jsea.2010.36061 Published Online June 2010 (http://www.SciRP.org/journal/jsea)

Testability Models for Object-Oriented Frameworks

Divya Ranjan[1], Anil Kumar Tripathi[2]

[1]Department of Computer Science, Faculty of Science, Banaras Hindu University, Varanasi, India; [2]Department of Computer Engineering, Institute of Technology, Banaras Hindu University, Varanasi, India.
Email: ranjan_divya@yahoo.co.in, aktripathi.cse@itbhu.ac.in

Received April 10th, 2010; revised April 21st, 2010; accepted April 23rd, 2010.

ABSTRACT

Frameworks are time-tested highly reusable architectural skeleton structures. They are designed 'abstract' and 'incomplete' and are designed with predefined points of variability, known as hot spots, to be customized later at the time of framework reuse. Frameworks are reusable entities thus demand stricter and rigorous testing in comparison to one- time use application. The overall cost of framework development may be reduced by designing frameworks with high testability. This paper aims at discussing a few metric models for testability analysis of object-oriented frameworks in an attempt to having quantitative data on testability to be used to plan and monitor framework testing activities so that the framework testing effort and hence the overall framework development effort may be brought down.

Keywords*: *Object-Oriented Frameworks, Complexity, Framelet-Based Design and Testability

1. Introduction

Frameworks represent semi-codes for defining and implementing time-tested highly reusable architectural skeleton design experiences and hence become very useful in development of software applications and systems. The object-oriented paradigm provides a promising set of solutions to attain reusability with the help of objects, classes, templates, inheritance, overloading and genericity [1,2] so object-oriented frameworks are much more com- mon than non object-oriented frameworks and have become a synonym for software frameworks. It should be noted that a framework without object-orientation is also possible. As per Gamma *et al*. [3], famous in reuse literature as *gang of four*, an object-oriented framework is a set of cooperating classes that make up a reusable design for a specific class of software which provides architectural guidance by partitioning the design into abstract classes and defining their responsibilities and collaborations. The five elements of an object-oriented framework, as identified by Gurp and Bosch [4], are design documents, interfaces, abstract classes, components and classes. An abstract class usually has at least one unimplemented operation deferred to for its implementation during framework reuse.

Applications are built from frameworks by extending or customizing the framework, while retaining the original design. The framework-centered development lifecycle broadly consists of following three phases [5]: 1) the **framework development phase** aims at producing reusable design in a domain, consisting of domain analysis, architectural design, framework design, framework implementation, framework testing activities; 2) the **framework usage phase** is also referred to as the framework instantiation phase or application development phase. Once framework is developed, it is deployed for the development of framework-based applications that include the core framework designs or part of it and new classes need to be developed to fulfill the actual applications requirements and 3) the **framework evolution and maintenance phase**.

One has to be very careful about developing *fault free reusable frameworks* because if the framework contains defects, the defects will be passed on to the applications developed from the framework during framework usage phase [6]. The reusable framework thus demands stricter and rigorous testing in comparison to a one-time use application [7,8]. It would be advisable to guaranty the production of high quality frameworks without incurring heavy costs for rigorous testing. This calls for analyzing testability of reusable artifacts so as to reduce the overall cost of framework-based development [9].

Bache and Mullerburg [10] define testability as the minimum number of test cases required to provide total

test coverage, assuming that such coverage is possible. Several definitions of software testability are available in literature but intuitively, a software component that is testable has the following desirable properties [11]:

- test sets are small and easily generated,
- test sets are non-redundant,
- test outputs are easily interpreted and
- software faults are easily locatable.

In spite of wide importance and promotion of frameworks, over the last decades, a widely accepted set of measures to quantify its characteristics has not been established. Moreover, there is a complete lack of framework testability metrics related studies in literature that could produce quantitative data on testability to be used to plan and monitor framework testing activities so that the framework testing effort and hence the overall framework development effort may be brought down. A recent work [9] proposed models for testability analysis of framework that particularly consider that the frameworks are inherently abstract and variable in nature. This paper proposes few more models for testability analysis of object-oriented frameworks, considering few design related aspects of frameworks.

Some obvious reasons for rigorous testability analysis of a framework could be summarized as [9]:

1) A testable framework ensures *low testing cost* and helps in reduction of overall development cost of a framework which has been designed and implemented as a semi-code.

2) Frameworks are reusable entities and hence high testability is essential. As testable system is known to provide increased reliability.

3) High testability brings *high reusability*. Many a times a framework reuser will want to test few features to assess its quality. If testing a framework is tough then framework reuser will hesitate in testing and using the framework and will seek to choose another framework or go for development without deploying a framework.

4) To calm obvious scientific curiosity that while writing test cases for frameworks why it is tougher in some case than the other cases or, so to say, why for one framework we had to think very hard before we were able to write a meaningful test suite, whereas for other frameworks we could generate test cases in a straightforward way.

5) Testability holds a prominent place as part of the *maintainability* characteristic in ISO 9126 quality model ISO, 1991, so this study also increases our understanding of software quality in general.

6) Framework testability analysis creates a base for formulating the strategy for designing highly testable frameworks, *i.e.*, *framework design for test* (FDFT).

The paper is organized in three sections. Proposed testability models for software frameworks appear in Section 2 and Section 3 presents conclusions.

2. Models for Testability Analysis of Object-Oriented Frameworks

This section aims at discussing various metric models for testability analysis of object-oriented frameworks considering few design related aspects of frameworks. Following discussion takes into account the factors that affect the testability of an object-oriented framework, as identified by Ranjan and Tripathi in [12]. They identified various factors and sub factors that affect the testability of frameworks so as to take care of those factors to bring high testability in frameworks. As per their observations, the factors that affect the testability of a framework are related to the characteristics of documentation of a framework, domain of a framework, design of a framework and the test support available for the framework testing like test tools, environments, reusable test artifacts and built-in tests etc.

2.1 A Testability Model Considering the Structural Complexity of a Framework

It is empirically proved that complexity metrics are good predictors of testing effort [13]. An interesting model of OO system complexity, proposed by Tegarden and Sheetz [14], consists of the *system complexity*, *structural complexity*, and *perceptual complexity constructs*. System complexity represents aspects of OO techniques and characteristics inherent to the problem. This construct influences the structural aspects of the system and the perceptual complexity of the OO system. Structural complexity deals with the measurable characteristics of the resulting OO system such as the number of classes and the interconnections between the classes whereas perceptual complexity deals with the ability of the developer to understand the problem, the structural components of the OO system, the use of OO techniques, and how to integrate these ideas to create an OO system.

As per this model, structural complexity identifies four levels of describing complexity of OO systems: variable, method, object, and system. At each level, measures are identified that account for the cohesion (Intra) and coupling (Inter) aspects of the system. Thus, the structural complexity of object-oriented framework may be expressed as:

$$SC_{Fr} \propto \sum_{i=1}^{N} CO_{Intrai} + \sum_{i=1}^{N} CO_{Interi} \quad (1)$$

where,

$$CO_{Intra} \propto CV_{Intra} + CM_{Intra} \quad (2)$$

and

$$CO_{Inter} \propto CV_{Inter} + CM_{Inter} \quad (3)$$

where,

N = Total number of objects in the framework
SC_{Fr} = Structural Complexity of the framework
CO_{Intra} = Intra Object Complexity in the framework
CO_{Inter} = Inter Object Complexity in the framework
CV_{Intra} = Intra Variable Complexity in an object
CV_{Inter} = Inter Variable Complexity in an object
CM_{Intra} = Intra Method Complexity in an object
CM_{Inter} = Inter Method Complexity in an object

It is very clear that framework testability is inversely proportional to its structural complexity. Therefore, using Equation (1) we can write,

$$Tb_{Fr} \propto \frac{1}{\sum_{i=1}^{N} CO_{Intrai} + \sum_{i=1}^{N} CO_{Interi}} \quad (4)$$

It for sure that $\sum_{i=1}^{N} CO_{Intra\,i}$ and $\sum_{i=1}^{N} CO_{Interi}$, in the above model, will never be zero together. Because total number of objects in the framework will always be >= 1. In case when N = 1, the value of $\sum_{i=1}^{N} CO_{Interi}$ may become zero but the value of $\sum_{i=1}^{N} CO_{Intra\,i}$ will not be zero then also.

2.2 A Testability Model Considering Complexity of Framework Interfaces

A framework may have its interfaces linked to external entities like other *frameworks*, *components*, or *library functions* etc. [15], known as external interfaces. More the number of other entities to be integrated with the framework, the more effort are required for their integration testing. This effort increases with the number and complexities of the constituent interfaces.

A framework testability model, considering complexity of framework interfaces, may be expressed as

$$Tb_{Fr} \propto \frac{1}{\left(1 + \sum_{i=1}^{N_{FItf}} C_{FItf\,i} + \sum_{i=1}^{N_{CInf}} C_{CInf\,i} + \sum_{i=1}^{N_{LInf}} C_{LInf\,i}\right)} \quad (5)$$

where,

C_{FItfi} = Complexity of framework's ith interface with other framework
C_{CInfi} = Complexity of framework's ith interface with component
C_{LInfi} = Complexity of framework's ith library interface
N_{FItf} = Total number of interfaces with other frameworks
N_{CInf} = Total number of interfaces with component
N_{LInf} = Total number of library interfaces

2.3 A Testability Model Considering Heaviness of Framework in Terms of its Size

Size happens to be an obvious influencing factor for testing effort [16]. A framework of huge size, *i.e.* consisting of large number of classes, methods, attributes and depth of inheritance etc., is considered heavy. We here make use of (a subset of) object-oriented metrics proposed by Chidamber and Kemerer [17].

The number and complexities of the methods involved in the framework is a predictor of how much time and effort is required to test. We define $WMFr$ (the consolidated weighted method per framework in line with WMC proposed in [17]) as follows:

$$WMFr = \sum_{i=1}^{Nc} WMC_i \quad (6)$$

where

$$WMC_i = \sum_{j=1}^{NMi} c_{ij} \quad (7)$$

and

$WMFr$ = Sum of weighted methods of all classes in FUT
WMC_i = Weighted method per class of ith class in FUT
Nc = Total number of classes in FUT
C_{ij} = Complexity of jth method in ith class of FUT
N_{Mi} = Total number of methods in ith class

Framework testability model considering heaviness of framework in terms of its size may be defined as follows:

$$Tb_{Fr} \propto \frac{1}{\sum_{i=1}^{Nc} (\sum_{j=1}^{NMi} c_{ij})} \quad (8)$$

because,

$$TE_{Fr} \propto WMFr \quad (9)$$

Nc and N_M appearing in Equation 8 may further be expanded as below in Equations 10 and 11

$$N_C = NOCC_{Fr} + NOAC_{Fr} \quad (10)$$

where,
Nc = Total number of classes in FUT
$NOCC_{Fr}$ = Number of concrete classes in the FUT
$NOAC_{Fr}$ = Number of abstract classes in the FUT

And, methods in a class shall comprise of all the *concrete* methods, *abstract* methods, and *overridden* methods. Thus,

$$N_M = N_{CM} + N_{AM} + N_{OM} \quad (11)$$

where
N_M = Total number of methods in FUT
N_{CM} = Total number of concrete methods in FUT
N_{AM} = Total number of abstract methods in FUT
N_{OM} = Total number of overridden methods in FUT

2.4 A Testability Model Considering Framelet-Based Design of Frameworks

Framelets are the small, flexible, relatively independent and reusable assets, which are designed based on the concept of frameworks. They are mini-frameworks that usually contain less than ten classes and have a simple, clearly defined interface [18]. They evolved as a means of modularizing the frameworks where a family of interrelated framelets is an alternative to complex frameworks. It is basically *a design principle* that instead of designing one full fledged and complex framework, design it with a family of related framelets with lean interfaces [19]. Designing a framework in framelet-based fashion promotes reducing overall complexity of the framework and is like using divide and conquers approach to facilitate both, the design and testing of the framework.

Understanding a framelet-based framework is easier than a full fledged and complex framework because a framelet-based framework is made up of loosely coupled small frameworks, known as *framelets*. Testability of a framelet-based framework depends upon the testability of constituent framelets and the coupling among them. So, we may write,

$$Tb_{Fr} \propto \sum_{i=1}^{N_{Fmlt}} \left(\frac{1}{COUP_{Fmlti}} \times Tb_{Fmlti} \right) \quad (12)$$

where

Tb_{Fr} = Testability of the framework

N_{Fmlt}= Total number of constituent framelets

$COUP_{Fmlti}$ = Sum of measure of coupling of ith framelet with other framelets

Tb_{Fmlti} = Testability of ith framelet

Since, a framelet is nothing but a small framework, consisting of not more than ten classes and very lean interfaces with other framelets [19], so any discussion or metric model regarding framework's testability will be applicable for estimating testability of a framelet.

Each of the testability metric models, discussed above, has different intention or applicability which is discussed in the following **Table 1**, however, more than one model may also be employed at the same time.

Table 1. Applicability of framework testability metric models

S.No.	Category	Framework Testability Metric Model	Applicability of the Model
1.	Testability models considering various kinds of complexities of frameworks	Testability Model Considering the Structural Complexity of a framework	When framework ***structural complexity*** is of concern for testability.
2.	-- do--	Testability Model Considering Complexity of Framework Interfaces	When ***framework integration*** with other frameworks, components and library functions is of concern for testability.
3.	Testability models considering size and framelet-based design of the framework	Testability Model Considering Heaviness of Framework in terms of its Size	When framework ***size*** is of concern for testability.
4.	-- do--	Testability Model Considering Framelet-based design of frameworks	When framework has ***framelet-based*** design and the testability of framelets are of concern.

3. Conclusions

It is quite obvious that the quality of the software system which has been developed with reuse depends heavily upon the quality of the underlying reusable artifacts. Frameworks are an important reusable artifact and are believed to be at the core of leading-edge software technology in the twenty first century [20]. Software frameworks and design patterns make reuse of design experiences possible but unlike design patterns (that may not have any code) software frameworks are semi-codes and hence call for thorough testing before they can be deployed as reusable entities. Although the technology for constructing frameworks and framework-based software is relatively advanced, we comparatively lack a sufficient theoretical basis for testing them. This paper attempted to discuss a few metric models for testability analysis of object-oriented frameworks in an attempt to having quantitative data on testability to be used to plan and monitor framework testing activities so that the framework testing effort and hence the overall framework development effort may be brought down. The framework testability models presented here takes into account few design related aspects of object-oriented frameworks.

REFERENCES

[1] J. W. Hooper and R. O. Chester, "Software Reuse: Guidelines and Methods," Perseus Publishing, Cambridge, 1991.

[2] M. Smolarova and P. Navrat, "Software Reuse: Principles, Patterns, Prospects," *Journal of Computing and Information Technology*, Vol. 5, No. 1, 1997, pp. 33-49.

[3] E. Gamma, R. Helm, R. Johnson and J. M. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software," Addison-Wesley Professional Computing Series, Massachusetts, 1994.

[4] J. Gurp and J. Bosch, "Design, Implementation and Evolution of Object-Oriented Frameworks: Concepts and Guidelines," *Software - Practice and Experience*, Vol. 31, No. 3, 2001, pp. 277-300.

[5] J. Bosch, P. Molin, M. Mattsson and P. Bengtsson, "Object-Oriented Framework-Problems and Experiences,"

Research Report, University of Karlskrona/Ronneby, Sweden, 1997.

[6] J. Al-Dallal and P. Sorenson, "System Testing for Object-Oriented Frameworks Using Hook Technology," *Proceedings of the* 17*th IEEE International Conference on Automated Software Engineering*, Edinburgh, September 2002, pp. 231-236.

[7] J. S. Poulin and J. M. Caruso, "Determining the Value of a Corporate Reuse Program," *IEEE Computer Society International Software Metrics Symposium*, Baltimore, May 1993, pp. 16-27.

[8] E. J. Weyuker, "Testing Component-Based Software: A Cautionary Tale," *IEEE Software*, Vol. 15, No. 5, 1998, pp. 54-59.

[9] D. Ranjan and A. K. Tripathi, "Variability-Based Models for Testability Analysis of Frameworks," *Journal of Software Engineering and Applications*, Vol. 3, No. 6, 2010, pp. 455-459.

[10] R. Bache and M. Mullerburg, "Measures of Testability as a Basis for Quality Assurance," *Software Engineering Journal*, Vol. 5, No. 2, 1990, pp. 86-92.

[11] R. S. Freedman, "Testability of Software Components," *IEEE Transactions on Software Engineering*, Vol. 17, No. 6, 1991, pp. 553-564.

[12] D. Ranjan and A. K. Tripathi, "Testability Analysis of Object-Oriented Frameworks," *The Journal of Defense Software Engineering*, accepted for publication.

[13] H. M. Olague, L. H. Etzkorn, S. L. Messimer and H. S. Delugach, "An Empirical Validation of Object-Oriented Class Complexity Metrics and their Ability to Predict Error-Prone Classes in Highly Iterative, or Agile Software: A Case Study," *Journal of Software Maintenance and Evolution*: *Research and Practice,* Vol. 20, No. 3, 2008, pp.171-197.

[14] D. P. Tegarden and S. D. Sheetz, "Object-Oriented System Complexity: An Integrated Model of Structure and Perceptions," Presented at *OOPSLA* 1992. http://www.acis.pamplin.vt.edu/faculty/tegarden/wrk-pap/OOPSLA92.pdf

[15] M. Mattsson and J. Bosch, "Framework Composition: Problems, Causes and Solutions," *Proceedings of Technology of Object-Oriented Languages and Systems*, Netherlands, 1997, pp. 203-214.

[16] P. Jalote, "An Integrated Approach to Software Engineering," Narosa Publishing House, Darya Ganj, 2009.

[17] S. R. Chidamber and C. F. Kemerer, "A Metrics Suite for Object-Oriented Design," *IEEE Transactions on Software Engineering*, Vol. 20, No. 6, 1994, pp. 476-493.

[18] W. Pree, "Design Patterns for Object-Oriented Software Development," Addison-Wesley Publishing Company, Massachusetts, 1995.

[19] W. Pree and K. Koskimies, "Framelets—Small and Loosely Coupled Frameworks," *ACM Computing Surveys*, Vol. 32, No. 6, 2000.

[20] M. E. Fayad and D. C. Schmidt, "Object-Oriented Application Frameworks," *Communications of the ACM*, Vol. 40, No. 10, 1997, pp. 32-38.

