International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:4, 2008

The Coverage of the Object-Oriented Framework Application Class-Based Test Cases

Jehad Al Dallal, and Paul Sorenson

***Abstract*—**An application framework provides a reusable design and implementation for a family of software systems. Frameworks are introduced to reduce the cost of a product line (i.e., family of products that share the common features). Software testing is a time consuming and costly ongoing activity during the application software development process. Generating reusable test cases for the framework applications at the framework development stage, and providing and using the test cases to test part of the framework application whenever the framework is used reduces the application development time and cost considerably.

Framework Interface Classes (FICs) are classes introduced by the framework hooks to be implemented at the application development stage. They can have reusable test cases generated at the framework development stage and provided with the framework to test the implementations of the FICs at the application development stage. In this paper, we conduct a case study using thirteen applications developed using three frameworks; one domain oriented and two application oriented. The results show that, in general, the percentage of the number of FICs in the applications developed using domain frameworks is, on average, greater than the percentage of the number of FICs in the applications developed using application frameworks. Consequently, the reduction of the application unit testing time using the reusable test cases generated for domain frameworks is, in general, greater than the reduction of the application unit testing time using the reusable test cases generated for application frameworks.

***Keywords*—**FICs, object-oriented framework, object-oriented framework application, software testing.

I. Introduction

An application framework provides a reusable design and implementation for a family of software systems [1]. It contains a collection of reusable concrete and abstract classes. The framework design provides the context in which the classes are used. The framework itself is not complete. Users of the framework complete or extend the framework to build their particular applications. Places at which users can add their own classes are called hooks [2].

Jehad Al Dallal is with Department of Information Sciences, Kuwait University, P.O. Box 5969, Safat 13060, Kuwait (e-mail: jehad@cfw.kuniv.edu).

Paul Sorenson is with Department of Computing Science, University of Alberta, Edmonton, AB. T6G 2H1, Canada (e-mail: sorenson@cs.ualberta.ca).

To build an application using a framework, application developers create two types of classes: (1) classes that use the framework classes and (2) classes that do not. Classes that use the framework classes are called Framework Interface Classes (FICs) because they act as interfaces between the framework classes and the second type of the classes created by application developers. Fig. 1 shows the relation between the framework classes, the hooks, the FICs, and the other application classes. FICs use the framework classes in two ways: either by subclassing them or by using them without inheritance. Hooks define how to use the framework and, therefore, they define the FICs and their specifications and show how to implement them.

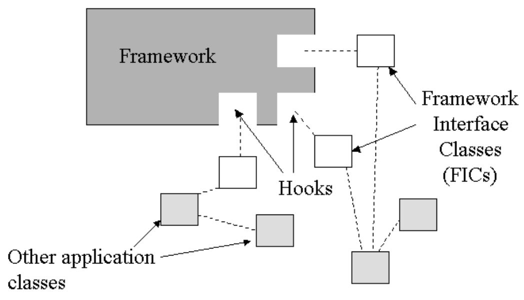

Fig. 1 Framework application classes

Frameworks are classified according to their scope into three types [3]: enterprise application frameworks, system infrastructure frameworks, and middleware integration frameworks. Enterprise application frameworks are also known as domain frameworks and they address different types of applications in a broad application domain such as telecommunications, avionics, manufacturing, and financial engineering. System infrastructure frameworks are also known as application frameworks and they address different types of applications in different application domains. Moreover, they simplify the development of portable and efficient system infrastructure including frameworks for user interfaces, communication frameworks, and operating systems. Finally, middleware integration frameworks are also known as support frameworks and they are used to integrate distributed applications and components. ORB frameworks, message-oriented middleware, and transactional databases are common examples for this type of frameworks.

Software testing is a critical and important stage of the application software development life-cycle and it affects the overall quality of the software. In a typical programming project, approximately half of the effort is spent on testing activities [4]. In object-oriented testing, each class in the system under test has to be tested individually. Class testing is a unit testing step with respect to application testing and the first level of integration testing. At class testing level, the method responsibilities, intraclass interactions, and superclass/subclass interactions are considered [5].

In [6], we have proposed a technique called all paths-state to build effective specification-based reusable unit level test cases for the FICs at the framework development stage. These test cases are provided with the framework to test the implementations of the FICs at the application development stage. In this paper, we measure the application unit testing cost reduction using the reusable test cases. The cost reduction is measured in terms of the number of implemented FICs in the applications and their total number of lines of code (LOC) in comparison to the total number of classes implemented at the application development stage and their total number of LOC. We count the number of LOC because it is a commonly used measurement for the size of code.

The case study is conducted using three frameworks: Client-Server Framework (CSF) [7], swing [8], and SalesPoint [9]. The former two frameworks are application frameworks while the later one is a domain framework. The results of the case study show that the percentage of use of the FICs in the applications constructed using the domain framework is, on average, much higher than the percentage of use of the FICs in the applications built using the application frameworks. As a result, the reduction in unit testing cost of the applications constructed using the domain frameworks is, on average, much higher than the reduction in unit testing cost of the applications built using the application frameworks.

The paper is organized as follows. Section II, introduces the used frameworks, discusses the case study settings, and shows the case study results. Section III discusses issues related to applications that use multiple frameworks. Finally, Section IV provides conclusion and discussion of future work.

II. Case Study

Thirteen applications developed using three different frameworks were considered in the case study. This section introduces the frameworks, illustrates the case study settings, and shows the results.

A. Used Frameworks

Applications developed using three frameworks were considered in the case study: CSF, Swing, and SalesPoint. CSF and Swing are application frameworks, while SalesPoint is a domain framework. CSF is a communications framework written in Java and developed to support the building of relatively small applications that require client-server or peer-to-peer communication support. CSF also provides persistent storage capabilities and can handle the communications over a TCP/IP connection using a model similar to email. CSF deals with synchronous and asynchronous messages sent between remote objects. The framework code consists of 38 classes and about 1.4K lines of code (without comments/blank lines). CSF hooks describe the behavior of ten FICs and show how they can be implemented or customized.

Swing is a Java framework developed to support GUI applications. In Java 1.3.1, Swing consists of 460 classes.

SalesPoint is a framework written in Java and developed to create point-of-sale simulation applications such as a ticket vending machine application or a big supermarket with many departments application. The framework supports the management of the relations between the business, the customers, and the administrative tasks like accounting. SalesPoint framework consists of 161 classes and the hooks describe the behavior of 78 FICs and show how they can be implemented or customized.

B. Case Study Settings

Performing the analysis required in this case study for relatively large number of applications requires exhaustive effort. Therefore, the case study was conducted using thirteen randomly selected applications out of a pool of 39 applications. Five of the applications use one framework and eight applications use two frameworks. As shown in tables 1-4, the applications use the following frameworks: one application uses CSF only, two applications use the Swing framework only, two applications use the SalesPoint framework only, four applications use CSF and the Swing framework, and four applications use the SalesPoint and Swing frameworks.

The CSF applications were developed by fourth-year undergraduate students at the University of Alberta. The SalesPoint framework applications were developed by second-year undergraduate students at the University of the Federal Armed Forces Munich. Finally the Swing applications were developed by a combination of the second and fourth-year undergraduate students, in conjunction with their application development activities on CSF and SalePoint.

For each application, the classes implemented at the application development stage were counted. The number of classes does not include the number of framework classes. In addition, the number of LOC of the counted classes is also counted. These two figures were counted using the

LOCC tool [10]. LOCC is a Java tool that produces size data corresponding to the number of packages, the number of classes in each package, the number of methods in each class, and the number of lines of code in each package, class, and method. The LOCC tool does not count comments and blank lines as part of the lines of code. Tables I, II, and III show the application name, the total number of application classes not including framework classes, and the number of lines of code (LOC) of each application. The FICs included in the applications were counted manually. Since the Swing framework has no associated hooks, we used the definition of the FIC to find the implemented FICs and count them. Every class in the considered applications that extends or uses a Swing class is an implemented FIC. For each application developed using the Swing framework, we counted the implemented Swing FICs. Finally, the total number of LOC for the FICs is the summation of the LOC of each of the FICs counted using LOCC tool.

C. Case Study Results

For each application, the first column of Tables I, II and III shows the name of the application. The second column shows the number of application classes not including the framework classes. The third column shows the number of LOC of the classes counted in the second column. The fourth column shows the number of FICs implemented in the application and the percentage of the number of FICs in the application. The last column shows the total number of LOC of FICs and the percentage of the number of LOC of the FICs in the application.

For applications developed using application frameworks, Table I shows that an average of 41.4% of the classes of the CSF applications are FICs. In terms of LOC, an average of 28.3% of the LOC of the CSF applications are for FICs. Table II shows that an average of 14.9% of the classes of the Swing framework applications are FICs. In terms of LOC, an average of 13.9% of the LOC of the Swing framework applications are for FICs. For applications developed using domain frameworks, much higher percentage averages were found. Table III shows that an average of 68.5% of the classes of the SalesPoint framework applications are FICs. In terms of LOC, an average of 75.5% of the LOC of the SalesPoint framework applications are for FICs.

TABLE I
APPLICATIONS DEVELOPED USING CSF

Application Name	Number of classes	Number of LOC	Number of FICs	Number of LOC in FICs
Student management system	47	3887	31 (66%)	1568 (40.3%)
Chatting system	55	7464	3 (5.5%)	179 (2.4%)
Course management system	44	3191	17 (38.6%)	667 (20.9%)
StoneClash Strategy Game	106	5324	56 (52.8%)	2050 (38.5%)
Army Game	149	8792	66 (44.3%)	3449 (39.2%)
Average	80.2	5731.6	**41.4%**	**28.3%**

TABLE II
APPLICATIONS DEVELOPED USING SWING FRAMEWORK

Application Name	Number of classes	Number of LOC	Number of FICs	Number of LOC in FICs
Hook Master	112	10520	9 (8%)	611 (5.8%)
Java Master	66	3846	4 (6.1%)	251 (6.5%)
Chatting system	55	7464	21 (38.2%)	2639 (35.4%)
Course management system	44	3191	7 (15.9%)	425 (13.3%)
StoneClash Strategy Game	106	5324	23 (21.7%)	2117 (39.8%)
Army Game	149	8792	15 (10.1%)	1797 (20.4%)
Tiler shop system	39	3114	4 (10.3%)	159 (5.1%)
Photo-service system	76	8831	10 (13.2%)	493 (5.6%)
Casino system	41	8859	2 (4.9%)	69 (0.8%)
Pizza shop system	59	4516	12 (20.3%)	182 (4%)
Average	74.7	6445.7	**14.9%**	**13.7%**

TABLE III
APPLICATIONS DEVELOPED USING SALESPOINT FRAMEWORK

Application Name	Number of classes	Number of LOC	Number of FICs	Number of LOC in FICs
Fast food shop system	18	1161	13 (72.2%)	890 (76.7%)
Tiler shop system	39	3114	28 (71.8%)	2174 (69.8%)
Photo-service system	76	8831	41 (53.9%)	6659 (75.4%)
Casino system	41	8859	25 (60.1%)	5042 (56.9%)
Golf club system	50	5041	45 (90%)	4821 (95.6%)
Pizza shop system	59	4516	37 (62.7%)	3534 (78.3%)
Average	47.2	5253.7	**68.5%**	**75.5%**

III. USING MULTIPLE FRAMEWORKS

When multiple frameworks are used to build an application, the number of FICs is equal to the summation of the number of FICs created using the hooks of each of the frameworks. In our case study, eight applications use two frameworks. Table IV shows the application names, used frameworks, the total number of FICs, and their total LOC. The last row of the table calculates the average of the total number of FICs by summing the percentages of the corresponding columns in Tables I, II, and III and dividing the result by the total number of summed percentages. The same calculation method is applied for the total number of LOC of the FICs in the last row of the table.

Table IV shows, not surprisingly, that the average of the total number of FICs counted by considering all the frameworks used in the applications is much higher than the average obtained by considering only one of the used frameworks for each application. A similar result is found for the total number of LOC in the FICs. This means that if an application uses multiple frameworks, considering the reusable test cases of all of the used frameworks in an application can reduce the class testing time more, on average, than considering the reusable test cases of one framework only.

TABLE IV
FRAMEWORK APPLICATIONS THAT USE MULTIPLE FRAMEWORKS

Application	Used frameworks	Total number of FICs	Total number of LOC in FICs
Chatting system	Swing & CSF	24 (43.6%)	2818 (37.8%)
Course management system	Swing & CSF	24 (54.5%)	1092 (34.2%)
StoneClash Strategy Game	Swing & CSF	79 (74.5%)	4167 (78.3%)
Army Game	Swing & CSF	81 (54.4%)	5246 (59.7%)
Tiler shop system	Swing and SalesPoint	32 (82.1%)	2333 (74.9%)
Photo-service system	Swing and SalesPoint	51 (67.7%)	7152 (81%)
Casino system	Swing and SalesPoint	27 (65.9%)	5111 (57.7%)
Pizza shop system	Swing and SalesPoint	49 (83.1%)	3716 (82.3%)
Average		**64.6%**	**63.2%**
Average using one framework		**32.8%**	**31.7%**

IV. CONCLUSION AND FUTURE WORK

In this paper, we conducted a case study to examine the reusability of the FICs in several framework applications. The case study shows that a high percentage of the classes of applications developed using domain frameworks are FICs, while the percentage of the FICs in applications developed using application frameworks varies largely according to the specification domains of the framework and the applications. The results support the hypothesis that the reusability of the FICs in the applications developed using domain frameworks is likely to be greater than the reusability of the FICs in the applications developed using application frameworks. At the framework development stage, reusable test cases can be generated for the FICs to be used at the application development stage. As the percentage of the FICs increases in the application, the part of the application tested using the reusable test cases increases and the amount of testing work required at the application development stage reduces.

Typically, building reusable test cases is a costly task. The case study results indicate that it is worthwhile to build reusable test cases for applications developed using domain frameworks as the original investment will be recouped after producing a few number of framework applications. However, it might not be worthwhile to build reusable test cases for some application frameworks because of the relatively low percentage of the FICs in the applications developed using the frameworks. Finally, in cases involving multiple frameworks, the case study results show that considering the reusable test cases provided with all the frameworks used in an application can save more testing time than using the reusable test cases provided with one framework.

Application developers can add new specifications to the FICs at the application development stage. These specifications are not covered by the reusable test cases built at the framework development stage. This means that the reusable test cases can cover part of the implemented FICs but not all. In our future work, we plan to study the percentage of the specifications of the implemented FICs covered by the reusable test cases. Our preliminary results show that, on average, a high percentage of the specifications (measured in terms of transitions in the state-transition models of the FICs) of the implemented FICs are covered using the reusable test cases.

References

[1] K. Beck and R, Johnson. Patterns generated architectures, Proc. of ECOOP 94, 1994, 139-149.

[2] G. Froehlich, H.J. Hoover, L. Liu, and P.G. Sorenson. Hooking into Object-Oriented Application Frameworks, Proc. 19th Int'l Conf. on Software Engineering, Boston, May 1997, 491-501.

[3] E. M. Fayad and D. C. Schmidt. Object-oriented application frameworks, Communications of the ACM, October 1997, Vol. 40, No. 10.

[4] K. Saleh, A. Boujarwah and J. Al-Dallal, "Anomaly detection in concurrent Java programs using dynamic data flow analysis", Journal of Information and Software Technology, Jan 2002, Vol 44, no 1, pp. 53-61.

[5] R. Binder. Testing object-oriented systems, Addison Wesley, 1999.

[6] J. Al Dallal and P. Sorenson, Generating Class-Based Test Cases for Interface Classes of Object-Oriented Black Box Frameworks, submitted for publication in Transactions on Engineering, Computing and Technology, 2006.

[7] G. Froehlich, Hooks: an aid to the reuse of object-oriented frameworks, Ph.D. Thesis, University of Alberta, Department of Computing Science, 2002.

[8] Java 1.3.1, http://java.sun.com/, July 2006.

[9] The SalesPoint framework v2.0 homepage, http://www-st.inf.tu-dresden.de/SalesPoint/v3.0/, July 2006.

[10] LOCC (software information), http://csdl.ics.hawaii.edu/Tools/LOCC/, July 2006.