Advanced Computing: An International Journal (ACIJ), Vol.3, No.4, July 2012

ANALYZING THE EFFICIENT TEST ORDER FOR INTEGRATION TESTING

Dr. Reena Dadhich[1] and Sourabh Sehgal[2]
Associate Professor, Govt. Engineering College, Ajmer (RTU, Kota) [1]
reena.dadhich@gmail.com[1]
Govt. Engineering College, Ajmer (RTU, Kota) [2]
reachsourabh@yahoo.co.in[2]

ABSTRACT

One major characteristics of Object Oriented Software is the complex dependency that exists between classes due to three different types of relationships that are inheritance, association and aggregation. Due to these dependencies one major problem arise while integrating and testing the object oriented software in order to reduce the number of required test stubs and to determine the test order for testing different classes. This paper presents a comparison between different test orders by exploiting a model produced during design stages (e.g. using UML), namely the Use Case Diagram and Class Diagrams. Our goal is to study and compare different test orders. Based on which we will propose efficient test order to reduce the number of stubs as well as time of testing. For the analysis of our proposed method we will take software developed for ATM machine.

KEYWORDS

Stubs, UML, Integration Testing, Test Order.

1. INTRODUCTION

Software testing is one of the most important activities in software development life cycle. Software organizations spend the large percentage of their budget in testing related activities of the developed software and ready to be implemented. A well tested software system will be validated by the customer before acceptance as discussed in [1]. The standardization of semi-formal modeling methods, such as UML reveals that testing can no longer be separated from specification/design/code stages: design-for-testability is a necessary basis for final-product reliability.UML is a widely accepted set of notations for modeling Object Oriented System, to build testable and thus, hopefully, trustable OO systems[2]. Use of UML diagrams helps a lot in communication of project teams to explore potential designs, and to validate the architectural design of the software. It has various diagrams for depicting the dynamic behavior of the objects in a system [18].

The most important objective of class integration testing is to find error(s) during interaction of classes. Therefore it is relevant to consider this interaction during integration process. If one class is integrated before another class on which it depends, a dummy class that full fills the behavior of the second class is needed. This dummy class is known as Stub as given in [14].

DOI : 10.5121/acij.2012.3405

Many OO Integration strategies have been proposed time to time in various research papers [5-7] and [9]. Other approaches addressing the class integration order problem have recently been proposed in research papers [8] and [9] during 2001-2003. These approaches are either based on Genetic Algorithms or graph-based approaches [3] and [9]. These strategies have two primary objective common points. The first most important objective is to minimize the number of required stubs. And the second objective is to determine the efficient integration order to reduce number of integration steps.

The order in which integration testing is performed is very important. Briand et al. [3] explained that in case of object oriented software, integration order of classes affects the efficiency and cost of testing. The test order is important for several reasons. First, the test order affects the order in which classes are developed. Second, inter-class test order impacts the use of test stubs and drivers for classes and the preparation of test cases. Third, inter-class test order determines the order in which inter-class faults are detected.

In Object Oriented Software all classes are integrated either from most dependent class to least dependent class or from least dependent class to most dependent class. In this paper we compare these two different test orders of integration testing.

The organization of the paper is as follow: In section 2 we will study the related work. In section 3 we will discuss some desirable properties for class integration. Then in section 4, we will compare different class integration order. Then in section 5 we will find the best testing order using example. In the whole paper we will take an example of ATM machine as given in [16] for our study and in section 6 the paper concludes with analysis about the efficient testing order.

2. RELATED WORK

Strategy used to integrate two existing methods aimed at breaking cycles so as to allow a topological order of class have been proposed by Tai et al and Le Traon [11]. A new class integration testing strategy based on a new Class Dependency Model has been proposed by Badri et al in [5]. Various approaches used for integration testing of Object Oriented applications that have been modeled in Unified Modeling Language are discussed by M. Waqas Raza [17]. Different model, strategy and methodology for planning integration and regression testing from an OO model have been proposed by Thierry Jéron and his team in [13].

3. TEST ORDER: PROPERTIES

In integration testing the test order is also referred as inter-class test order. In this section we will discuss about the different properties for inter-class test order. In our case study model i.e. the software for ATM machine, there is a class ‘ATM’ and we will take this class as a testing class.

To find the efficient testing order there are some desirable properties. To understand these properties we assume that testing a class ‘ATM’ (as shown in Fig: 1 Object Relation Diagram (ORD) for ATM) for interclass Integration involves the following two steps:

i) Outgoing edge of 'ATM' must be tested at least once. If 'ATM' has an outgoing edge to class 'CardReader' that has not been integrated yet, then a stub for 'CardReader' is used for testing this edge.

ii) Each incoming edge of 'ATM' from a class that has been integrated, retest this edge at least once. Such an edge was tested earlier using a stub for 'ATM'. Retesting such an edge is needed since a stub for 'ATM' is a simplified version of 'ATM'

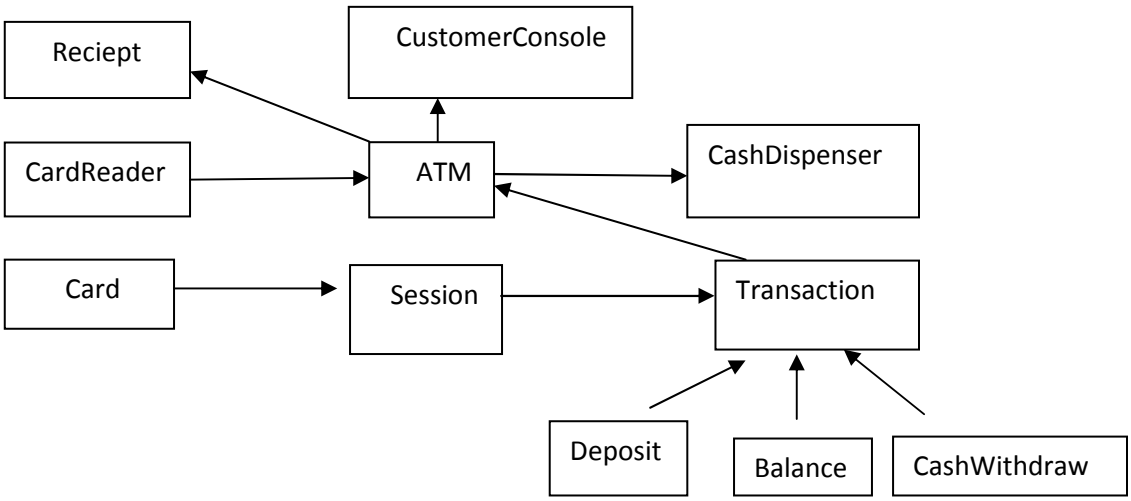

Fig 1: ORD (ATM)

If class 'ATM' has an outgoing inheritance or aggregation edge to class 'CardReader', then 'CardReader' should be tested before 'ATM' for inter-class integration. Since inheritance and aggregation relations between classes do not form cycles, the following properties for inter-class test order are desirable:

Property 1: For any two classes shown in Fig: 1 ORD say 'CardReader' and 'ATM', if there exists a directed path from 'CardReader' to 'ATM' such that the path contains inheritance and aggregation edges only, then 'CardReader' is tested before 'ATM' for inter-class integration.

As shown in Fig: 1 an Association edge exists between these two classes, so if we test 'ATM' class first then we need stub of 'CardReader' class. This property can also be extended by allowing association edges.

Property 2: For classes 'ATM' and 'CardReader' in an ORD shown in Fig 1, if there exists a directed path from 'CardReader' to 'ATM' and no directed paths from 'ATM' to 'CardReader', then 'CardReader' is tested before 'ATM' for inter-class integration.

If association edge contains in ORD, then ORD must contains cycles. To remove cycles in ORD, we must remove association edges. Then only we can produce test order. One of the major issue in integration testing is creating stubs for those classes that are still not integrated. The number of stubs can be reduced based on the association edge we are deleting.

Property 3: ORD that contains cycles, reduce the number of stubs needed for integration testing at reasonable cost. The second issue occurs when a class 'ATM' whose stub was used earlier is being tested for inter-class integration. As mentioned earlier, all incoming association edges of 'ATM' from classes that have been integrated need to be retested. Now the question arises in what order we should test these association edges?

For this we consider two classes say 'Receipt' and 'CashDispenser' have outgoing association edges to 'ATM' and 'CashDispenser' is integrated before 'Receipt'. The association edge from 'CashDispenser' to 'ATM' should be retested before that from 'Receipt'' to 'ATM', since there may exist a directed path from 'Receipt'' and 'CashDispenser' that contains inheritance and aggregation edges.

PROPERTY 4: Suppose that classes 'Receipt' and 'CashDispenser' have outgoing association edges to 'ATM', 'CashDispenser' is tested for inter-class integration before 'Receipt', and 'Receipt' is tested for inter-class integration before 'ATM'. When 'ATM' is being tested for inter-class integration, the association edge from 'CashDispenser' to 'ATM' should be retested before that from 'Receipt' to 'ATM'.

4. TEST ORDER: COMPARISON

Test order is an decisive factor of test work, it is a most valuable factor that working out a high-efficiency test order of classes in Object Oriented Software to minimize test work [15].

So one of the most important tasks of Object Oriented Integration testing is to find the efficient test order in which all classes are integrated [19]. Basically there are two types of integration order i.e. from most dependent class to least dependent class and from least dependent class to most dependent. The test order used for class integration affects the number of stubs, efforts and time required for testing. In the next section we will analyze these two orders in respect of efforts and time required for testing, and by comparing results of two we can propose efficient test order which will give integrated classes.

5 CASE STUDY

In some papers [3, 10, 11, 12], the testing effort is estimated by counting the number of stubs that are required for testing by assuming that all stubs are equally difficult to write. When the number of stubs increases testing efforts increases proportionally and vice versa.

SCENARIO 1: We are considering Banking ATM example and test all classes in two different orders from least dependent class to most dependent class and most dependent class to least depend. For this we are using class diagram for an ATM machine [16] in banking sector which is modeled through Unified Modelling Language (UML).

The basic structure of the class diagram arises from the responsibilities and relationships discovered when doing the CRC cards and Interaction Diagrams. As we estimate efforts with number of stubs [3], [10] and [12], we consider one constant value X, Let's say X=10.

Based on the class diagram shown in Fig 2, we have constructed a Table 1 which shows different attributes, methods and dependencies between two classes. The table also shows the data related to the number of stubs as well as estimated efforts required for a particular class.

It is clear from the table that different classes have different number of dependencies. As the number of dependency increases, number of stubs also increases. So now the question arise which class should we test first, the class having low dependency or the class having high dependency? This can be compute with the estimated efforts. Estimated Efforts mentioned in Table 1 is calculated on the basis of

Number of Stubs * X (constant).

Thus as the number of stubs increases, the testing efforts increases. Thus we can conclude that if we test less dependent class then the estimated effort is less as compared to the case when we test more dependent class first. We can also say that testing effort is directly proportional to the number of stubs. So testing should start from the class that require least number of stubs and so on. This is a better order to test the classes.

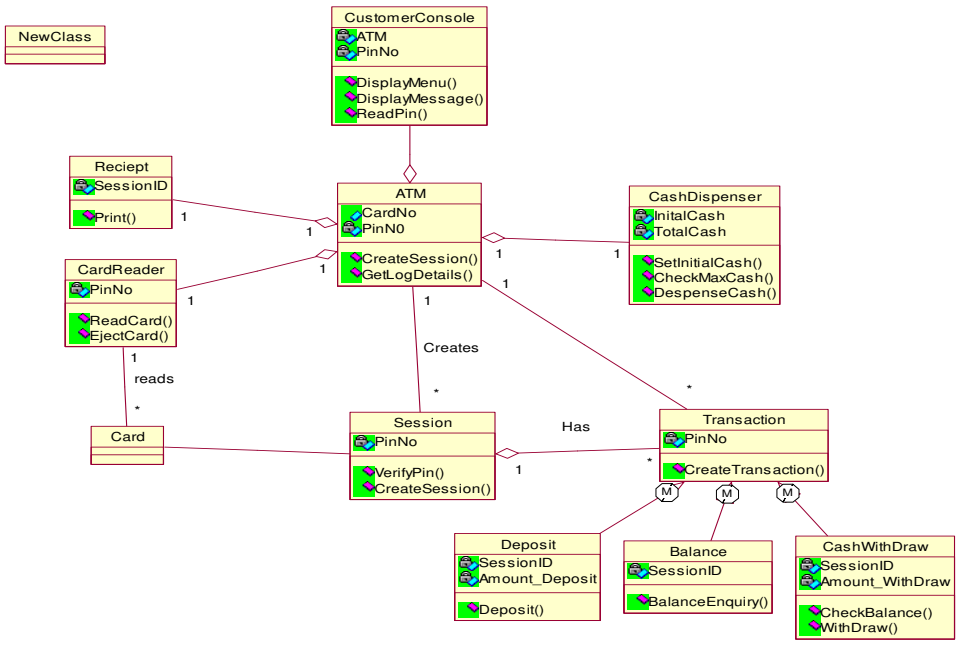

Fig 2: Class Diagram (ATM)

SCENARIO 2: Now we are considering another scenario i.e. time factor to compare testing order. Let us suppose each class takes T seconds to test. And t seconds are needed to build one stub.

Considering Table 1, the class 'ATM' require 7 stubs, as this class is most dependent class then total time to test this class is (T+7t) where as the classes 'Cash Dispenser' and 'Operator' need only one stub thus the total time required for testing each of these classes is (T+ 1t).

As the number of dependencies increases, number of required stubs increases and ultimately testing time increases. Thus if we test more dependent class first, testing time increases a lot. From this we can say that if we start testing from least dependent class, the testing time decreases as the number of required stubs decreases.

6. CONCLUSION

From both above mentioned scenarios as mentioned in section V we can conclude that test order from least dependent class to most dependent class is much efficient than the test order from most dependent class to least dependent class as both the time and efforts increases when we start testing from most dependent class and move towards the least dependent classes.

Table 1: Class name, attributes, and methods for a Class Diagram

Class Name	**Attributes**	**Methods**	**Number of Dependencies**	**Number of Stubs**	**Estimated Effort**
ATM	atmid:integer, bankname:string, state:string,location:string	performStartup() ,performShutdown() createSession(),getLogDetails()	7	7	70
CustomerConsole	atm:ATM	displayMenu(), displayMessage() readPIN()	1	1	10
CardReader	atm:ATM	readCard(), ejectCard()	2	2	20
CashDispenser	Initialcash:integer, totalcash:integer	setInitialCash(), checkMaxCash() dispenseCash()	1	1	10
Operator	atm:ATM	switchOn(), switchOff(), checkATMStatus()	1	1	10
Session	atm:ATM,pin:integer,state:string	createSession(),verifyPIN()	3	3	30
Transaction	atm:ATM, session: Session, pin:integer,	createTransaction()	2	2	20

	balance:integer				
Deposit	amount:integer, bankname:string , pin:integer	getDetails(),perf ormDeposit()	2	2	20
Withdrawal	amount:integer, bankname:string , pin:integer, balance:integer	getDetails(),,perf ormWithdrawal ()	2	2	20
BalanceEn quiry	pin:integer, bankname:string	getDetails(),perf ormEnquiry()	2	2	20

REFERENCES

[1] Thierry Jéron, Jean-Marc Jézéquel, Yves Le Traon, and Pierre Morel IRISA-INRIA," Efficient Strategies for Integration and Regression Testing of OO Systems", Campus Universitaire de Beaulieu, 35042 Rennes Cedex, FranceThierry.Jeron, Jean-Marc.Jezequel, Yves.Le_Traon, Pierre.Morel}@irisa.fr.

[2] Clay E. Williams, "Software testing and the UML", International Symposium on Software Reliability Engineering (ISSRE'99), Boca, Raton, 1999.

[3] L.C Briand , Labiche and Y Wang, "Revisiting Strategies for Ordering Class Integration Testing in the Presence of Dependency Cycles", Proc' 12"' ISSRE,2001

[4] F. Gavril, "Some NP-complete Problems On Graphs", Proc. 1977 Conf. on Information Sciences and Systems, April 1977, pp. 91-95.

[5] Mourad Badri, Linda Badri and Soumia Layachi, "Vers une stratedie de tests unitaires et d'integration des classs dansles applications orient'ees object " , Revne Genie Logiciel, N. 38, 1995.

[6] A. Bertolion, P. Inverardi, H. Muccini, A. Rosetti, "An approach to integration testing based on architectural descriptions", Proc. of the Third IEEE International Conference on Engineering of Complex Computrt Systems, 1997, pp.77-84.

[7] Rober V. Binder, "Design for testablity in OO systems", Communication of ACM, 1994, Vol. 37, pp. 87-100.

[8] L. Briand, J. Feng and Y. Labiche, "Experimentiong with Genetic Algo and Coupling Measures to Devise Optimal Test orders", Software Engineering with computational Intelligence Kluwer, 2003.

[9] V. Le Hahn, K. Akif, Y.Le Traon and J.M. Jezequel, "Selecting an efficient OO integration testing strategies", 15th European Conference for OO programming, Budapest, June 2001.

[10] D. Kung, J. Gao, P. Hsia, Y. Toyoshima, and C. Chen. "A test strategy for object-oriented programs" . In 19'h Computer Software and Applications Conference (COMPSAC 95), 244, Dallas, TX, August 1995. IEEE Computer Society Press, pages 239.

[11] K.C. Tai and F. Daniels. "Test order for inter-class integration testing of object-oriented software". In The Twenty-First Annual International omputer Software and Applications Conference (COMPSAC '97), Santa Barbara CA, 1997. IEEE Computer Society, pages 602-607.

[12] L. C. Briand, Y. Labiche , and Y. Wang. "An investigation of graph-based class integration test order strategies". IEEE Transactions on Software Engineering, , July 2003, pages: 29(7): 594-607.

[13] Bertolino.A, "Software Testing: Guide to the software engineering body of knowledge", IEEE Software, Vol. 16, 1999, pp. 35-44.

[14] Linda Badri, Mourad Badri and Velou Stephane Ble- Department of Mathematics and computer Sccience University of Quebec at Trois-Rivieres.,"A Method Based Approach for OO Integration Testing: An Experimental Study."

[15] Q'an Chen (Department of ComputerScience XiaMen UniversityXiamen 361005 China), Xiaojiang Li (Department of TestEngineering The Academy of Equipm ent C & T Beijing 101 41 6 China) –"An Order-Assigned Strategy of Classes Integration Testing Based on Test Level".

[16] Rajni Pamnani, Pramila Chawan, Satish Salunkhe Department of computer technology, VJTI University, Mumbai- "Object Oriented UML Modeling for ATM Systems"

[17] M. Waqas Raza Computer Science Department Mohammad Ali Jinnah University Islamabad, Pakistan-"Comparison of Class Test Integration Ordering Strategies", IEEE - International Conference on Emerging Technologies, September 2005, 17-18, Islamabad.

[18] F. Basanieri and A. Bertolino. "A Practical Approach to UML-based Derivation of Integration Tests". In Proc. of the 4th International Quality Week Europe QWE2000.

[19] Q. Chen and X. Li, "An Order-Assigned Strategy of Classes Integration Testing Based on Test Level", IEEE 2003.

Authors:

Sourabh Sehgal Sourabh Sehgal had done his B.Tech(CS) from Ch. Devi Lal Memorial Engineering College Panniwala Mota Sirsa (Haryana). At present he is doing M.Tech (Software Engineer) from Rajasthan Technical University, Kota India. He has 2 year of Software Industry Experience. He had presented paper in conferences.

Dr. Reena Dadhich is presently working as a Associate Professor and Head of the Department of Master of Computer Applications at Engineering College Ajmer, India. She received her Ph.D. (Computer Sc.) and M.Sc.(Computer Sc.) degree from Banasthali University, India. Her research interests are Algorithm Analysis & Design ,Wireless Ad-Hoc Networks and Software Testing. She has more than 12 years of teaching experience. She is working as an Editorial Board Member/Reviewer/Committee member of various International Journals and Conferences. She has written many research papers and authored as well as edited many books.