Volume 3, Issue 4, April 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Testing Techniques for COTS – A Survey

Jagdeep singh *CSED, Thapar University, India.*
Shvivani Goel *CSED, Thapar University, India.*

***Abstract*— The use of Commercial of the Shelf (COTS) based product in the software development has been proposed a way of reducing both development and implementation cost. There are a large number of COTS products available in market. The features provided by COTS and cost associated with them is an important consideration in selecting COTS. So there is a need of evaluation of COTS products before procuring them for software reuse. COTS testing is best way of evaluating the COTS products. In this paper, a survey of various techniques used for testing COTS is presented. The techniques discussed are software wrapping, compatibility and regression testing, contract based mutation, boundary value and partition testing, and interface mutation testing.**

***Keywords*— COTS, software testing, interface mutation testing, contract based mutation**

I. INTRODUCTION

The use of COTS product has been increasing in recent year, but their usage has resulted in various new problems. First, their source code is rarely available. Even if the source code is available, how a COTS component will behave after integration is still a challenging issue[1]. There are many other issues related with COTS, the interfaces and their formats, component interaction, compatibility and integration. There are numerous testing techniques proposed for testing of COTS product. Jennifer et al. have proposed software wrapping technique in which the component is wrapped with glue code and all input and output at interfaces are recorded to understand component behaviour [2]. Leonardo et al. have proposed a technique for compatibility and regression testing of COTS-based software that can automatically generate compatibility and prioritized test suite based on behaviour models that represent component interaction [3]. Ye Wu has proposed black box testing technique based on partition and boundary value analysis [4]. Marcio et al. have proposed interface mutation, in which they proposed an approach for integration testing [5]. Ying et al. have proposed contract based mutation technique for testing components and it employs high level contract mutation operators [6]. The following sections describe all these techniques.

II. SOFTWARE COMPONENT TESTING TECHNIQUES

We have identified various COTS testing techniques that are suitable for testing software components. All of these are best suited for a particular scenario. The techniques considered for COTS testing are software wrapping technique, compatibility and regression testing technique, boundary value and partition testing, interface mutation testing and contract based mutation. These are described in the following sections in detail.

A. Software wrapping technique

This approach helps to gain an understanding that how a COTS component interact with the rest of the system. The key goal of this strategy is to deal with COTS component without any dependence on vendor. In this technique there is a layer called a wrapper that encapsulate the component. They key idea is to understand whether COTS component is interacting with the system or not. Various faults can be injected at the input to see the whether the component detects the faulty input. Thus predicting the reliability of software components becomes easy. The wrapper can also be accompanied by various mechanisms like monitor, records so that inputs and outputs of the component can be recorded for further observation. The key advantage of this approach is not dependent on vendor for evaluating COTS.

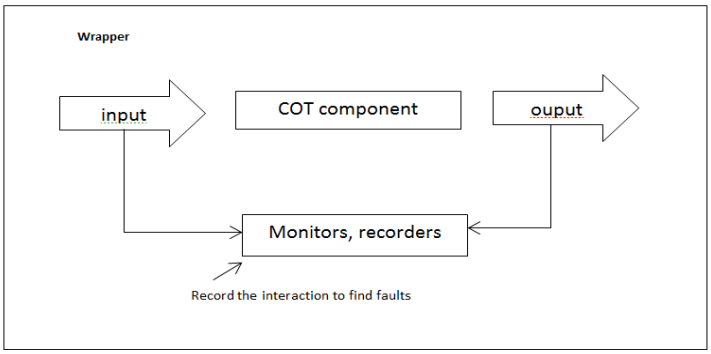

Fig1. Conceptual view of a wrapper

B. Compatibility and Regression Testing of COTS

The data exchanged between software components and system is of great use to identify potential faults. This technique first generates compatibility and regression test suites based on data exchanged between components and sequence of invocation to component's interfaces. Compatibility test suite helps in identifying and discarding the software components that are compatible with specifications. Regression test suite helps to improve the efficiency of regression testing and helps us to identify the potential problems that can occur at integration. This technique is based on inter-component behavior model, and these model are automatically derived while testing the previous version of software.

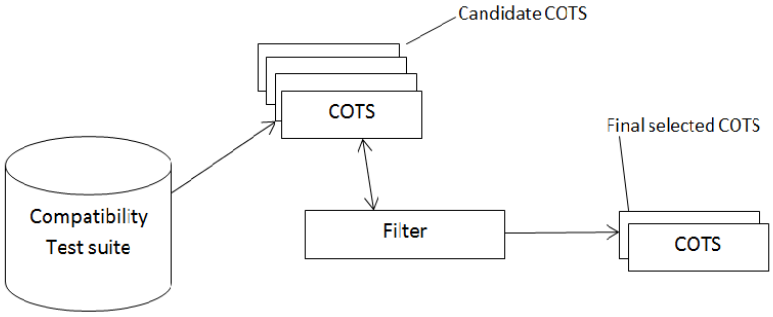

Fig. 2 Usage scenario of compatibility test suites

Compatibility and regression test suites are generated from behaviour models and further behaviour models are generated by behaviour capture and test (BCT) technology [7]. BCT is a dynamic analysis technique for generating behaviour models. Models are produced by recording the data value exchanged between the component and sequence of invoke methods. BCT automatically infer input, output and interaction models for generating behaviour models.

C. Boundary Value Analysis and Partitioning Testing

In COTS based system all the interaction are done through interfaces. So interfaces are the only point of contact and they play a key role in testing COTS components. Equivalence partition testing tries to divide the input domains into k different disjoint partitions p_1, p_2, …,p_k, where $p_1 \cup p_2 \cup \ldots \cup p_k$ = input domain, and $p_i \cap p_j = \Phi$ for any i and j where $i \neq j$. Values from each partition have "all or none" property. If an element is selected for a component and that element fails, then all elements from that partition will also fail. Since formal specification usually don't exist for COTS components, so there is no systematic way to generate equivalence partition. If formal specification exists, then partition can be automatically derived.

But usually faults occur at boundaries, so boundary value is key technique that can expose faults at testing [8]. Exhaustive testing is not possible as we cannot execute all the test cases to cover every state. Partition testing reduces the test cases and boundary value analysis address test effectiveness, so boundary value testing strategy must be used with partition. By analysis of various inputs at interface level of COTS, boundary value analysis technique can be applied to better understand the COTS component.

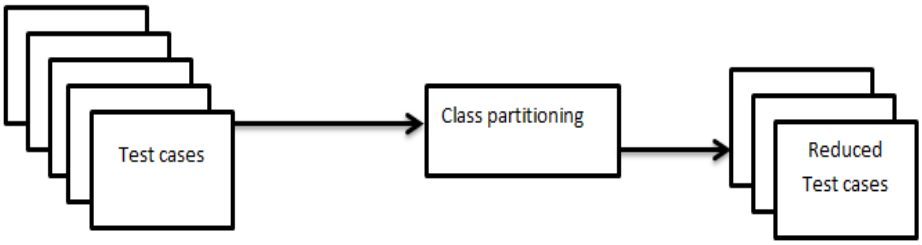

Fig.3 partition reduces the test cases

D. Contract Based Mutation for Testing COTS

COTS components follow standardized interfaces provided by various component models (COM, CORBA, JAVA BEANS) for interaction with system and providing various services. Mutation testing which is proposed by DeMillo[9] and Hamlet [10] is the widely accepted criteria in seeding faults for evaluating the quality of software. Ying et al. [6] have proposed contract based mutation that is based on idea to apply mutation testing to the contracts supplied with the components. Since contracts can be supplied without source code, mutating on the contract does not depend upon vendor anymore. This approach does not use the tradition mutation operator, instead mutation is applied on high level contract of the component.

Design by contract (DBC) is a method to improve software testability and the main motive is to establish contract between the provider and user of software component. Contracts define the behavioural feature of component, so if the contracts are violated while running, faults can be revealed. DBC usually helps in enhancing the testability and also the implementation of component.

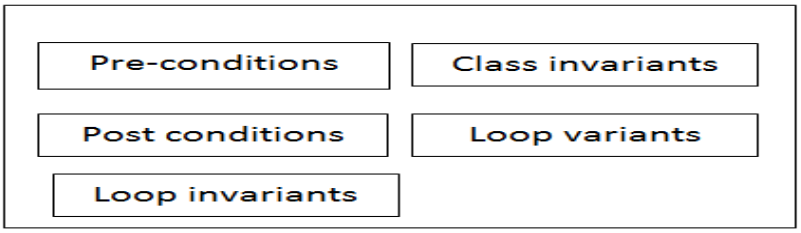

Fig.4 Typical structure of contract's

Typically, mutants are produced from any program by making small changes to the program. Following type of mutants can be produced from contracts. These are contract negation, condition exchange, precondition weakening, post condition strengthening and many more. Mutation score can be calculated like the traditional software testing. The idea of calculating mutation score remains the same as in traditional software. First calculate the number of mutants killed and number of total mutants created. The formula has been show below.

Mutation score= $D/(M_c-E_c)$

D= number of killed mutants
M_c= number of contract mutants
E_c = number of equivalent mutants

Higher the number of mutation score, more reliable is the COTS component.

E. Interface Mutation for integration testing

Interface mutation is proposed to evaluate the interaction between various modules. It helps to evaluate the test cases for a subsystem which consist of two or more components. The key idea of interface mutation is to test interaction between units and apply mutant operator at their interfaces such as function calls. The goal of integration testing is to put the units in their intended environment and exercise their interaction completely.

Various types of techniques have been proposed for testing inter-procedural test set. Haley et al. have classified integration error as computational error and domain integration error [11]. Linnenkugel et al. have defined control flow and data flow-based criteria [12]. Jin and Offut have proposed coupling based testing technique [13].

Interface mutation differs from the above approaches. The use of interface mutation helps us to explore the adequacy criteria at integration level. The nature of data exchanged form unit a to unit b can be categorized into four ways.

1. Data can be passed from unit a to unit b via input parameter (pass by value).
2. Data can be passed from unit a to unit b or returned to via input/output parameters (pass by reference).
3. Data can be passed by global variables
4. Data can be passed through return variables

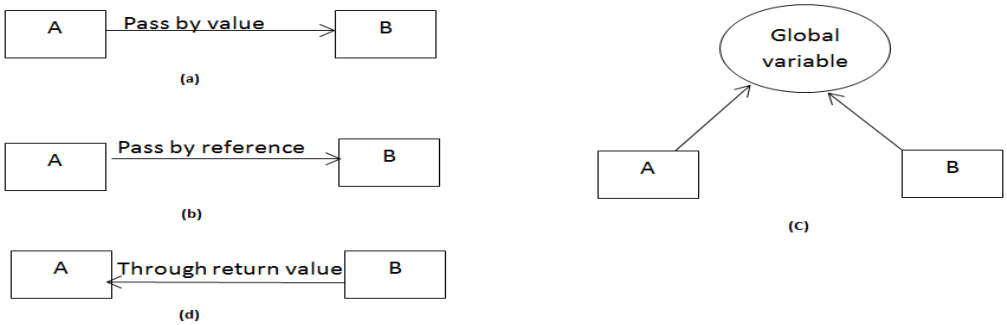

Fig.5 data exchange between two units

The next step is to generate mutants, but mutants in this technique differ from traditional software because small changes have to be made at interface level or connection between two units. Various sets of mutation operator can be applied to generate wide variety of mutants and check adequacy of test sets.

III. COMPARISON

Black box testing techniques are applied when source code of COTS is not available, i.e. when the COTS product are reused. White box testing techniques are applied when the source code of COTS products is available i.e. when while developing the COTS product. A comparison of various testing techniques with their key advantages and their limitations has been given in table 1 which also suggests the testing technique suitable for a particular scenario.

Table 1. Comparison of COTS testing techniques

Testing Technique	**Best Suitable for**	**Advantages**	**Limitations**
Software wrapping technique	Blackbox testing of COTS components	No more dependence on vendor for testing of COTS component.	It can record only behaviour of software component based on input given and output received.
Compatibility and regression testing	Regression testing of COTS components	It makes selection of software components by potentially eliminating software components that are not compatible with the specifications.	Regression test suites are time consuming and resource consuming.
Boundary value and equivalence class testing	Best suited when source code of COTS components are available	Boundary value testing helps to identify potential problems that occur at boundaries of data types (like integer ranges 0 to 32767) and equivalence class helps in reducing test case as exhaustive testing is not possible.	Cannot address all the software components testing issues because most of the times source code is not available.
Contract based mutation testing	Integration testing used when no source code of COTS components is available	Capable of testing interfaces of COTS that follow standardized component models (COM,DCOM,CORBA) through which COTS components communicate and provide services.	Just testing the interfaces through which software components communicates and provides the services does not solve the issue related with using software component when there is no source code available
Interface mutation testing	Integration testing when source code of COTS components is available	Capable of finding faults that can occur when components communicate with each other and the system. It addresses the entire integration problem that might exist when software component is integrated into the system.	This technique addresses the integration issues of software components but software components are still not tested. This technique is dependent on input output models and interaction between components.

IV. **Conclusion**

In this paper we surveyed a number of techniques that can be used for testing COTS components. Major problem with COTS component is that the source code is not available which makes their testing and evaluation a tough task. Software wrapping technique is capable of recording the input, output of software components so that fault can be identified and software components can become more and more reliable. Compatibility and regression testing solve the issues of selection of COTS components when source code is not available. Boundary value analysis address the issues that can happen at the boundaries because many faults occur at boundaries like array gets out of bound, range of an integer gets exceeded etc. Exhaustive testing is always not possible, so equivalence class partitioning helps us reduce the test cases. Interface mutation testing and contract based mutation; both address issues that can occurs at integration of software components. Though most of these techniques that we surveyed do not need the source code of COTS and they are not even dependent on vendor, still there is need of more techniques that can exploit the full potential of software components and promise for better future can be accomplished, where COTS components can be changed like electronic circuits.

REFERENCES

[1]. J.C. Knight, R. W. Lubinsky, J. McHugh, and K.J. Sullivan, "Architectural approaches to information survivability,"Technical Report CS-97-27, University of Virginia, Sept. 1997.

[2]. Haddox, J.M. and Kapfhammer, G.M., "An approach for understanding and testing third party software components", Reliability and Maintainability Symposium, 2002, pp. 293–299.

[3]. Leonardo Mariani, Sofia Papagiannakis, and Mauro Pezze, "Compatibility and Regression Testing of COTS-Component-Based Software" In *Proceedings of the 29th international conference on Software Engineering* (ICSE 2007), IEEE Computer Society, Washington, DC, USA, pp. 85-95.

[4]. Ye Wu,www.tuisr.utulsa.edu/iwicss/Black-box_Testing_for_Evolving_COTS-Based_Software.pdf, last accessed 11 march,2013.

[5]. Máarcio E. Delamaro, José C. Maldonado, and Aditya P. Mathur, "Interface Mutation: An Approach for Integration Testing", *IEEE Transaction on Software Enggineering* 27, 3 ,March 2001, pp. 228-247.

[6]. Ying Jiang, Shan-Shan Hou, Jinhui Shan, Lu Zhang,Bing Xie,"Contract-Based Mutation for Testing Components", ICSM, 2005, pp. 483-492.

[7]. L. Mariani and M. Pezz`e. Behavior capture and test: Automated analysis of component integration. In *proceedings of the IEEE International Conference on Engineering of Complex Computer Systems*, 2005.

[8]. Hoffman, D., Strooper P. and White, L., "Boundary Values and Automated Component Testing", *Journal of Software Testing, Verification and Reliability*, Vol. 9, No 1, 1999, pp. 3-26.

[9]. T.A. Budd, R.J. Lipton, F.G. Sayward, and R.A. Demillo, "The design of a prototype mutation system for program testing", In *Proceedings of the National Computer Conference*, Anaheim, CA, 1978, pp. 623-627.

[10]. J.M. Haddox, G.M. Kapfhammer, and C. C.Michael, "An approach for understanding and testing third party Software components", In *Proceedings of the Annual Reliability and Maintainability Symposium*, Seattle, WA, USA, 2002, pp. 293-299.

[11]. A. Haley and S. Zweben, "Development and Application of a White Box Approach to Integration Testing," The Journal of Systems and Software, vol. 4, pp. 309-315, 1984.

[12]. U. Linnenkugel and M. MuÈ llerburg, "Test Data Selection Criteria for (Software) IntegrationTesting," Proc. First Int'l Conf. Systems Integration, Apr. 1990, pp. 709-717.

[13]. Z. Jin and A.J. Offut, "Integration Testing Based on Software Couplings", Proc. 10th Ann. Conf. Computer Assurance (COMPASS '95) , Jan. 1995, pp. 13-23.

