Based on Markov Process Method for Integration Testing

Liu Shu-Fen¹, Zhang Yi-Kun², Zhao Jing¹, Xu Wen-Yuan³
¹College of Computer Science and Technology, Henan Polytechnic University, Jiaozuo 454000, China
liusf@jlu.edu.cn, zhaojing8384@163.com
²College of Computer Science and Technology, Jilin University, Changchun 130012, China
³System Engineering Research Institute, Beijing 100094, China
equalparadise@eyou.com

Abstract—As software systems increase in size, the heterogeneous software, in particular which is based on SOA, has gradually become a mainstream. But the number of methods which be used in integration testing of heterogeneous software is less. In this paper, it promotes a new integration testing method which is based on Markov process, according to the characteristics of Markov process, it calculates the optimal path of integration testing, and finally based on this model to generate the algorithm of test cases. The innovation of this method is that combining the data flow with control flow and be used in integration testing of heterogeneous software.

Index Terms—SOA, Markov, Integration Testing

I. INTRODUCTION

With the complexity of the software system, software engineering will increase with the external development in particular application software component integrated together to build software system, become the current trend of software development. Development of the traditional use of centralized software development platform, development tools, operating system architecture of the tight coupling, makes the physical dispersion gradually formed an independent system of called "information island." Service Oriented Architecture, SOA is a component model, it features the application of different units – services, and links through well-defined interfaces between services and contracts. Interface defines a neutral way, independent of the specific implementation services of hardware platforms, operating systems and programming languages, construction of such a system makes the services available in uniform and standard way to communicate. SOA with its strong autonomy, loosely coupled, coarse-grained, open, integrated, interoperable, extensible and other features are gradually gained broad industry support and academic attention.

Because of the characteristics of integrated software components, its conduct of the test methods are different to traditional testing methods, it has brought new problems and challenges to test theories, test architecture, and test implementation. This paper will combine the characteristics of Markov process, propose a model based on Markov process for heterogeneous software integration testing, and generate test cases.

II. SOFTWARE INTEGRATION TESTING

In 1979, Glemford J. Myers in the “The art of software testing” had been defined software test as follows: Software testing is the process of implementation of procedures or system in order to find errors. The integration testing is the next stage of unit testing, is the tested unit module assembled into systems or subsystems, tested again, and focus on test different parts of the interface module. Integration testing is used to check each unit module can combine together to work, and normal operate.

Component-based software design is the mainstream of current software development methods. This approach is characterized by large particle size enables software reuse, plug and play, reducing software development and maintenance costs. The system developed in this way is characterized by different components in different programming language development, some need to run on different operating platforms, the physical location is very far away, some components are developed independently, while other components are developed by third parties or commercial components, many components of the source is not in public. Thus, component-based software components on the existence of a so-called trust problem, effective way to address this problem is the software integration testing.

Internationally, the component tests began in the late 90s, in recent years, with the software to increase the size and complexity, component integration testing has become a hot topic. Currently, The integrated test method has been proposed mainly based on the state test, event-based test, based on faulty testing and forms and based on semi-formal description test. Each has advantages and disadvantages, state-based testing method is based on the FSM model generated test cases, this method is easy to cause an explosion which can not control the state. Event-based test is describing the synchronization sequence of events among the objects, generate test cases, but this method does not take into account the state
of the object changes, the test data is not sufficient. In the component software integration testing, more representative studies, including the Unified Modeling Language (UML) integration testing technology\(^6\), Component Interaction Test (CIT)\(^7\), sequence generation technique\(^8\), component interaction graph (CIG) model\(^9\), etc. Among them, UML-based component integration testing techniques have a good support system, it can be automatically generated test case sequence from the systematic point, But this test in the case of larger scale systems will inevitably produce explosive conditions portfolio problem; CIG will shift the focus to components and component interaction issues, but the method of pre-construction plans for CIG system needs to pay a high price.

In 2007, Li Changqing and other is based on Web Services of SOA software feature, and make full use of testing in the field of knowledge and methods, on this basis was proposed based on XML technology, the core algorithm for automated testing engine\(^1\). But the lack of how to generate test cases and test more effective. Liu Yulin and so on based on OOPN (object-oriented Petri net) integration testing\(^2\), is a data flow and control flow of the combination of test methods, but did not propose how to automate the tests. 2005, Jiang Ying proposed a WSDL-based document under the automatically generated using random test data\(^3\), technology options under the contract variation and then test data, but did not mention how to generate test cases test data, and the test cases automatically. Guo Yong in 2006 proposed building the concept of Web Services Testing Framework\(^4\), and how to use SOA methods to test Grid software, but did not adopt effective ways to implement test cases and analyze the test results. Lin Zhen proposed functional specifications based on the structured test methods, the method is based requirements specification and design specifications, generate test cases and test data sets, however, the method for structured programming more effective, but for Object-oriented and service-oriented software design is not suitable. Peking University Professor Yang Fuqing, Professor Mei Hong as early as 1995 proposed the use of formal methods to describe the components and assembly relationships. Renhong Min, Fudan University in 2003, Formal methods used are described on the composite component, focuses on the relationship between the component assembly mechanism and behavior deduced, But the lack of scholars component assembly method applied to the test area. Integration Testing of some other component developers need to test support, Some need to extract the source code from the component test the required information, there are way too theoretical and lacking practical application.

Because of some of the problems above, this paper presents a model based on Markov process.

III. MARKOV INTEGRATION TESTING MODEL

Tested for the whole software systems, assume that the first one has M components, path contains N. Our goal is to test all components and all the defects, and to exclude all of the defects. In the software testing process, if each time point (chosen path time) \(t = \{0, 1, 2, \ldots\}\) the number of remaining on the path of the system as \(t\) time in which the state \(S\), then the whole state space \(S = (S_t, t \geq 0) = (N, N-1, \ldots, 1, 0)\), each time point \(t\) according to some probability of choosing a path as an action, each time according to the different probability to selection paths as test the composition of all the action space \(A = (a_t, t \geq 0) = \{1, 2, \ldots, K\}\) and each time point \(t = \{0, 1, 2, \ldots\}\) on the state \(S_t\) and the actions \(a_t\) will affect the state of the next time \(t+1 S_{t+1}\). Therefore, the entire testing process is forming a Markov testing process, shown in Figure 1.

In order to facilitate the use of Markov processes for software testing process, without loss of generality, Make the following assumptions:

- **a)** Software in the initial time \((t = 0)\) contains a components of M and paths of N.
- **b)** Decision-making moments in each take only one action, and this action can only be detected up to a path.
- **c)** If at time \(t\), the system state is \(S_t = j\), the action applied to testing at the probability \(\theta_0\) to detect a path, the next time \(t+1\) the state \(S_{t+1} = j - 1\); with probability \(1 - \theta_0\), cannot detect the path, the state of the next time \(t+1\) is named time \(t\), that is, \(S_{t+1} = S_t \theta_0\) is action \(a_t\) could be interpreted as the each remaining path of software so that due to the probability of being detected, is called path detection rate.
- **d)** Once the path is detected, the path is considered to have been detected.
- **e)** \(S_t = 0\) is an ending state, which has completed testing of all paths.
- **f)** Take action \(a_t\) in each time \(t\), whether or not to detect the path will lead to the size of \(C_S(a_t)\) the test costs, at the end of the state to take any action on the testing costs are recorded as 0.
- **g)** at each time \(t\) has total K feasible action, action space \(A = \{1, 2, \ldots, K\}\).

Can see from the above assumptions, the Markov process is describing the software testing process can be defined as a five-tuple \((T, S, A, P, C)\), where, \(T = \{0, 1, 2, \ldots\}\) is Decision time, \(S = (N, N-1, \ldots, 1, 0)\) as state space, \(A = \{1, 2, \ldots, K\}\) as the action space, \(P\) for the transition probability matrix, its elements satisfy the Markov measure:

\[
P(S_{t} | S_{t-1}, a_{t-1}) = P(S_{t} | S_{t-1})
\]  

(1)
That at time t-1 action \( a_{t-1} \), the probability of shift from state \( S_{t-1} \) to state \( S_t \), which marks \( S_0, S_1, \ldots, S_n \), \( S_{t-1} \) expressed in the notation from state before the beginning of time, denoted by the empty set \( \varnothing \). C elements in the \( C_\tau(S_t(a_t)) \) that the test cost when the state of \( S_{t-1} \) action \( a_{t-1} \), from the initial time 0 to time \( t \), the whole testing process can be described to a Markov process history:

\[
H_t = \{S_0, a_1, S_1, a_2, \ldots, S_{t-1}, a_t, S_t\}
\]

Note \( \pi = (P(a_t | H_t); t = 0,1,2, \ldots) \) is the probability set which under the conditions of history \( H_t \) to take action \( a_t \), \( t = 0,1,2, \ldots \), that a testing strategy. Recorded \( \tau \) is the first time from the system state \( S_0 = N \) to reach state 0, for taken the minimum number of path testing operation, that is \( \tau = \min (\tau : S_\tau = 0) \), from time 0 to time \( \tau \) for the selected action forms a test case sequence \( X = (a_0, a_1, \ldots, a_\tau) \). Definition:

\[
\phi(X) = E_\pi[\sum_{t=0}^{\tau} C_s(a_t)]
\]

For the expected test cost, which \( E_\pi \) that demand relative to expectations of probability measure \( \pi \).

Our problem is to find an optimal test strategy to make the minimum expected test cost, that determine the optimal sequence of test cases, so:

\[
\min \phi(X) = E_\pi[\sum_{t=0}^{\tau} C_s(a_t)]
\]

Can be seen from the above, Markov testing process (or software testing strategy) decision \( a_t \) time \( t \) relative to the action of state of \( S_t \). For a given test target, An optimal software testing strategy should be able to design or to determine which action must be objective and relative test is optimal. This makes these actions \( (a_t, t = 0,1,2, \ldots, \tau) \) of the selected probability of a given test target relative to the formation of an optimal test profile.

### IV. Based on MITM Model Test Algorithm

As the MITM main task of integration testing is to discover the errors that may occur in the processing of message passing between components, test case is actually the message path mean as selection sequence, this path through the MITM model for integration testing. Algorithm described below:

First, implementing all components, recording all the path of components from the start to the termination.

Determining Markov process of five-group, and calculated according to model an optimal test strategy.

In accordance with the testing strategy for integration testing, and record the test results.

Figure 2, in an isomerization of the software system, Contains three components 8 path, as each action can only choose a path, Thus, according to the path detection rate to determine the path each time to choose. Suppose the path detection rate is \( \theta_1 = 0.1429, \theta_2 = 0.1667, \theta_1 = 0.2, \theta_2 = 0.25, \theta_3 = 0.3333, \theta_4 = 0.5, \theta_5 = 0.5, \theta_6 = 0.075, \)

\( \text{action space } A = \{1,2,3,4,5,6,7,8\} \), select A1 to B1 path the probability that action \( a_1 \) is \( p_1 = 0.125 \). And so, choose the path e of B1 to C2 the probability that action \( a_2 \) is \( p_2 = 0.5 \), choose C2 to B1 path f is the probability that action \( a_3 \) is \( p_3 = 0.2 \), select B1 to C1 of the path d is the probability of action \( a_4 \) is \( p_4 = 0.2 \), choose the path b A2 to B2 is the probability that action \( a_5 \) is \( p_5 = 0.25 \), choose the path c B2 to A3 is the probability that action \( a_6 \) is \( p_6 = 0.5 \), choose the path of g B2 to C3 that probability of action \( a_7 \) is \( p_7 = 0.5 \), choose the path of C4 to B2 h is the probability that action \( a_8 \) p8 = 1. This can be calculated an expected cost \( \psi (X) = 63.032 \). Similarly, because each choice of path is random, Therefore, the test fee for each is different, can calculate all the paths of the operation has completed each test cost. Therefore, the final test to select the lowest-cost actions that space, this constitutes the best test case. When choosing the path that the order (a, d, e, f, b, g, h, c), it tests the lowest cost, that is, \( \psi (X) = 44.570 \).

### V. Summary

According to the experimental results show that: The test engine than the previous state diagram based on UML, UML collaboration diagram based methods such as integrated test, better able to discover errors as soon as possible, location positioning error, analysis of error types, improve test coverage and test accuracy.

MITM take into account both the state and data mobility, but also considered the message is sent between objects and synchronization behavior sequence, Is a data flow and control flow testing methods combined. Future plans to achieve this model, a test tool above, and the basis of this model can automatically generate test cases above.

### REFERENCES


