IJCSC International Journal of Computer Science and Communication Vol. 2, No. 1, January-June 2011, pp. 1–10

TESTING OF DATABASE APPLICATION USING AGENT BASED METHOD

Jyoti Duhan

Department of Computer Science, Singhania University, Rajasthan, India,
E-mail: Jyotiduhan09@gmail.com.

ABSTRACT

Software testing is indispensable for all software development. As all mature engineering disciplines need to have systematic testing methodologies, software testing is a very important subject of software engineering.
Database systems have important and wide popularity among the world. It is common for software applications written in an imperative language to have access to the database through SQL statements embedded in the code. Testing of such can be manual, automated, or a combination of both. Agent technologies facilitate the automated software testing by virtue of their high level decomposition, independency and parallel activation.

This paper, deals with the process of testing a Database, through automated testing using multi-agents which will ensure that testing time is not wasted and aids in better decision-making for the Test Managers. We have proposed an effective automated test framework using multi-agents that manages testing of database applications thereby reducing the various resource attributes such as people, cost, time during the test process. This framework also ensures quality in the testing process by reducing the manual work.

We have measured the effectiveness of the test process through various metrices that enhances the quality of the process. We have implemented the test process for six different database applications and their effectiveness computed through metrics.

Keywords: Database Testing, Test Cases, JADE, Multi Agents, Coverage Tree Metrics, Command Form Metrics.

1. INTRODUCTION

Software test automation has long been recognized for its potential to improve the breadth of testing by reducing redundant, manual testing and maximizing repeatability and test accuracy. However, high costs and maintenance have plagued these traditional approaches. The efficient and cost effective approach to database test automation helps ensure that database applications meet the performance, functional and service-level expectations of users, customers and business partners. Automated testing does the checking of database predefined functions and reduces the time of database management.

Database systems have major importance and wide popularity among the software industry. Database applications are becoming very complex. Most of the database applications are subject to frequent change. The automation of database application testing can be broadly partitioned into the problems of test cases generation, test data preparation and test outcomes verification [13]. Conventionally, database application testing is based upon whether or not the application can perform a set of predefined functions. It is common for software applications written in an imperative language to have access to the database through SQL statements embedded in the code. These queries are part of the application's business logic. Because of this, it is necessary to have conducted suitable testing in the same way of the rest of the code. The tests should cover all the query situations and avoid producing undesired results so as to obtain their maximum possible coverage.

Agents can be delegated to perform simple tasks or to provide autonomous services to assist a user without being "commanded". Agents play roles in terms of different expertise and functionalities while supporting services to an application [27]. They are responsible for making intelligent decisions to execute such activities and to provide valid and useful results. Agents can work alone, but most importantly, they can cooperate with other agents. Agents are software components in the system that are viewed as many individuals living in a society working together.

Thus managing the testing of database application becomes a tedious task. To ease the job of the tester this project proposes an effective automatic test framework that manages testing of database applications thereby reducing the various resource attributes such as people, cost, time during the test process. This framework also ensures quality in the test process by reducing the manual work. It also measures the effectiveness of the test process through various metrics that enhances the quality of the test process. We have implemented the test process for six different database applications and their effectiveness computed through metrices.

The overall goal of this paper is to increase the efficiency of database testing and automate the testing process using Multi-Agents thereby easing the work of

This document was created using SOLID CONVERTER PDF To remove this message, purchase the product at www.SolidDocuments.com

the Tester and measure the effectiveness of the test process through various metrics that enhances the quality of the test process.

In Section 2 review of the related work in database testing is detailed. Section 3 describes the introduction to agents. In Section 4 describes the database testing tool. In Section 5 we have described problem statement. Section 6 describes the database testing with multi-agents. Section 7 details the Software Metrics. Section 8 describes Coverage Metrics for a Database application. Section 9 describes Experimental Databases used for Coverage Metrics Computation. Section 10 describes Result Analysis. Section 11 gives the conclusion and future work.

2. RELATED WORK

Researchers have considered various techniques for database application testing, and coverage metric for the queries. Researchers have considered various techniques for database application testing, and coverage metric for the queries. Yuetang Deng et. al. [9], in his work, have developed a tool named AGENDA that creates temporary tables to store the relevant data and converts the assertion into a check constraint at attribute/row level. He has considered static constraints involving aggregate functions, multiple tables, and dynamic constraints involving multiple database states in his execution [9].

As an extension of, Yuetang Deng et.al.'s work, A (test) GENerator for Database Application is proposed as a research prototype tool set for testing database application programs. In testing such applications, the states of the database before and after execution play important role along with the user's input and outputs [4].

Design mechanisms to create the deterministic rule set, non-deterministic rule set, and statistic data set for a live production database was proposed by Xintao Wu et. al. [6]. A Security Analyzer together with security requirements (security policy) and output were also built. The mock database generated from the new triplet was able to simulate the live environment for testing purpose, while maintaining the privacy of data in the original database [6].

Data flow testing [8] proposed by S.K. Gardikiotis et. al. involved generating test data to force execution of different interactions between variable definitions and variable references or uses in a program variable. Here Database applications were reverse engineered in order to facilitate the embedded SQL statements. The derived code contains calls to SQL modules stored in the database server. To test these modules, data flow analysis was provided with respect to the statements of data manipulation language [8].

In Gregory M. Kapfhammer, Mary Lou So®a work, a family of test adequacy criteria can be used to assess the quality of test suites for database driven applications [7]. Develop a unique representation of a database-driven application that facilitates the enumeration of database interaction associations. These associations can reflect an applications definition and use of database entities at multiple levels of granularity. The usage of a tool to calculate intra procedural database interaction associations for two case study applications indicates that our adequacy criteria can be computed with an acceptable time and space overhead [11].

In [1], dataflow and control flow analysis and the dependences between components of a database application were used to determine the components that should be tested when any change were produced and to minimize the set of test cases during regression testing, but its aim was not to design test cases.

In [15], an automated tool that provides the measurement of the coverage of SQL queries was presented.

In [14], William G.J. Halfond and Alessandro Orso computed the testing requirements and an e±cient technique for measuring coverage of these requirements was detailed.

In [23], concerning multi agent systems (MASs), some characteristics such as agent autonomy and asynchronous message-based interaction bring a degree of non-determinism which presents new testing challenges. JAT, a framework for building and running multi agent systems (MASs) test are in scenarios, which relies on the use of aspect-oriented techniques to monitor the autonomous agents during tests and control the test input of asynchronous test cases [23].

In most of the above works, some particular constraints of a database were selected and tested. None performed a complete framework for testing database application and proved the effectiveness of testing. In this paper, we have proposed an effective automated test framework that manages testing of database applications using multi-agents thereby reducing the various resource attributes such as people, cost, time during the test process. This framework also ensures quality in the test process by reducing the manual work. The effectiveness of this test process is measured through various metrics that enhances the quality of the process.

2.1 Intelligent Agents

The general definition of an intelligent agent is:

"A computer system that is situated in some environment and that is capable of flexible autonomous action in this environment in order to meet its design objectives". Intelligent agents possess certain key characteristics:

- Ability to communicate with other agent.
- Can operate without direct user intervention.
- Monitor environment and can directly affect surroundings.

This document was created using SOLID CONVERTER PDF To remove this message, purchase the product at www.SolidDocuments.com

- Ability to perform intelligent reasoning about events in environment.
- Learn from the events occurring in environment to better achieve design objectives.

2.2 Multi Agent System

The critical difference between multi-agent systems and individual agents focuses on the patterns of communication. A multi-agents system communicates with the application and the user, as well as with the other agents in the system to achieve their objectives. However, in the individual agent, communication channels are only open between the agent and the user. The key characteristics in multi-agent environments are:

- Multi-agent systems provide the infrastructure for inter-agent communication.
- Multi-agent systems are usually designed to be open concept without any centralized designer.
- The agents within a multi-agent system are autonomous and may be cooperative or competitive in nature.

The most important aspect of multi-agent systems is the communications between the agents. Many protocols have been developed that give the agents the ability to both receive and send information to each other.

2.3 Agents and Agent Algorithm

The easiest approach to the design of the multi-agent system is to a design an interface agent for each of the tasks. In order to incorporate the information gathered from the agents and send it to the software maintainer, another agent is required in the system design.

In the design of the algorithms for each of the interface agents, the first consideration was to build them to work by themselves and perform their designated function. Once these functions are in place, the multi-agent communication functionality would be built into the design.

2.4 User Interface and Multi-Agent System Communication

The user interfaces designed for this research are simple, but informative. The first agent will have interaction with the other agents. The other agents will be designed to provide information to the other agents. Upon completion of individual agent design, the task of creating the multi-agent communication protocol will be performed. From the research completed on multi-agents-communication protocol, the possible choices for use are:

- Knowledge Query and Manipulation Language (KQML).
- Foundation of Intelligent Physical Agents Agent Communication Language (FIPA ACL).
- The Agent Communication Language (ACL) is used for developing Agents and JADE is used for Java Programs.

3. DATABASE TESTING TOOLS

3.1 DbUnit

Database testing is performed efficiently using the tool DbUnit. DbUnit is an open source Framework created by Manuel Laflamme. This is a powerful tool for simplifying Unit Testing of the database operations [11]. It extends the popular JUnit test framework that puts the database into a known state while the test executes. This strategy helps to avoid the problem that can occur when one test corrupts the database and causes subsequent test to fail. DbUnit provides a very simple XML based mechanism for loading the test data, in the form of data set in XML file, before a test runs. Moreover the database can be placed back into its pre-test state at the completion of the test.

3.2 JADE

JADE (Java Agent Development Framework) is a software framework fully implemented in Java. It allows reducing the time-to-market for developing distributed multi-agent applications by providing a set of ready and easy-to-use functionalities that comply with the standard FIPA specifications and a set of tools that supports the debugging and monitoring phases [30].

FIPA is the international standardization body for software agent technologies. JADE has been widely used over the last years by many organizations (both industrial and academic) in a variety of contexts: research projects, industrial prototyping, tutorials, and as a teaching support for agent-related courses in many Universities all over the world. JADE is extremely versatile and its features and footprint can also fit the constraints of environments with limited resources.

4. PROBLEM STATEMENT

The problem addressed in this paper is as follows.

- Automate the process of generating test database, test case generation and test case execution.
- Define a measurement of coverage for SQL SELECT queries in relation to a database loaded with test data that can be used as an adequacy criterion to carry out the testing of applications with access to databases.

5. DATABASE TESTING WITH MULTI-AGENTS

The agent based database testing(ABDT) proposed here is to offer a definition for encompassing to cover the database testing phenomena based on agents, yet sufficiently tight that it can rule out complex systems that are clearly not agent based. Agent technologies facilitate the automated software testing by virtue of their high level

This document was created using SOLID CONVERTER PDF To remove this message, purchase the product at www.SolidDocuments.com

decomposition, independency and parallel activation. We have proposed framework for Agent based database testing is shown in Fig. 1.

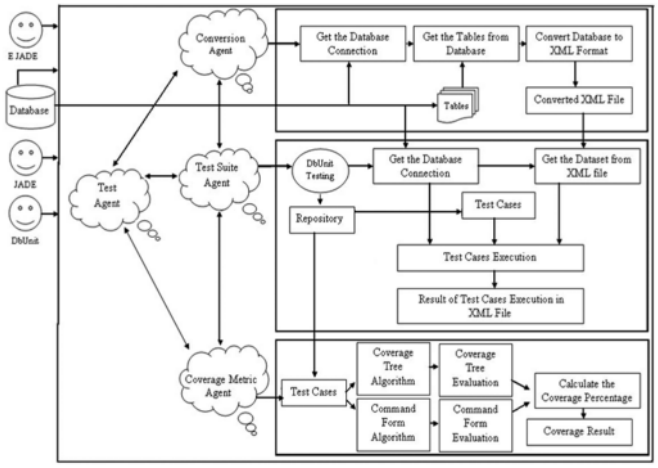

Fig. 1: Framework for Multi-Agent Based Testing of Database Application

In the framework the database to be tested is converted to an XML file by conversion agent and the dataset is got by the getDataSet method. Database connection is done by getconnection method. Using these methods all the test cases that are placed in the repository is executed and the actual output is compared with expected output for correctness. The valid test cases from the repository are used for calculating the various metrices to achieve quality. As this is done automatically by the framework, manual work is reduced, thereby reducing the resource attributes.

5.1 Test Agent

Test agent is the main agent that coordinates the conversion agent and test suite agent. The test agent activates each agents depending on the messages received from the agents and activates the other agents accordingly. During database testing, the test agent initiates the conversion agent first and checks whether the table conversion (Database table to XML file) has been completed first. After the completion of the work by the conversion agent, it activates the test suite agent. The test suite agent executes the test cases for the database from repository. After the completion of the test case execution, the test suite agent response to the test agent through a message. The test agent initiates the coverage metric agent which measures the effectiveness of the test process through various metrics that enhances the quality of the test process. The functionality of the test agent is shown in Fig. 2.

5.2 Conversion Agent

A Block Diagram for converting database into xml file for database testing using conversion agent is shown in Fig. 3.

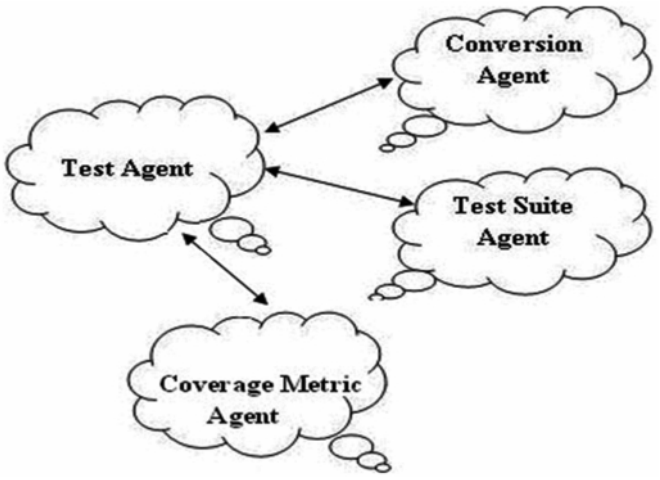

Fig. 2: Test Agent

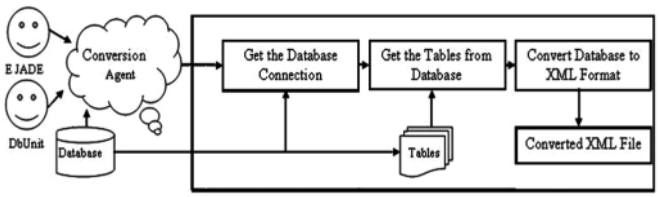

Fig. 3: Conversion Agent

Testing the database requires that the data should be in a known initial state. The database is converted as a XML data set for easy understanding of the DbUnit. The conversion agent is activated through the message from the test agent.

After the intimation of the test agent, the functionality of the conversion agent as follows:

- Checks for the oracle database connectivity.
- From the database it gets the table to be converted.
- Convert the table into an XML file.

If the table is not available it gives the message to the test agent. Otherwise the table is converted using the DbUnit by conversion agent. It converted the table from the database into an XML file.

Element names match table names and the attribute names match columns. Then it responds to the test agent by the completion message and terminates from the agent main container.

5.3 Test Suite Agent

The Block Diagram for Test suite agent execution for database testing is shown in Fig. 4.

Test suite agent gets the control from the test agent. The test suite agent functionality as follows:

- Checks for the oracle database connectivity.
- Gets the converted input xml file through getDataset () method.

This document was created using SOLID CONVERTER PDF To remove this message, purchase the product at www.SolidDocuments.com

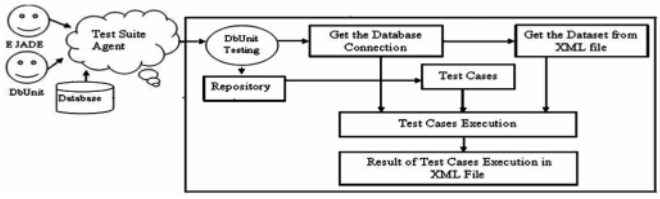

Fig. 4: Test Suite Agent

- Executes the test cases from the repository.
- Store the result of test case in output xml file.

For example the TestInsert Test Case. This operation inserts the dataset contents into the database. This operation assumes that table data exists in the target database and fails if this is not the case. To prevent problems with foreign keys, tables must be sequenced appropriately in the dataset. The testInsert test case gets the input from the xml and it inserts this into the database by executing the database operation INSERT.

5.4 Coverage Metric Agent

The Block Diagram for Coverage Metric Agent for database testing is shown in Fig. 5. Coverage metric agent gets the control from the test agent.

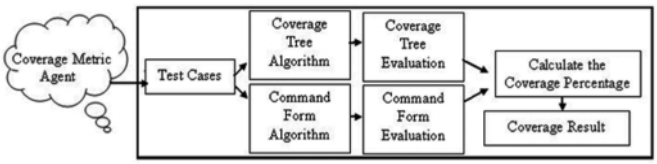

Fig. 5: Coverage Metric Agent

The Coverage metric agent functionality as follows:

- Initiates the coverage tree metric.
- The coverage tree metric calculates the coverage percentage for SELECT queries.
- After the completion it reports to the coverage metric agent.
- Then initiates the command form metric.
- The command form metric calculates the coverage percentage for SELECT queries.

6. SOFTWARE METRICS

Software metrics is defined as the current state of art in the measurement of software products and process [17]. Measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined unambiguous rules. Metrics strongly support software project management activities mainly test management. They relate to the four functions of management as follows:

(*i*) *Planning:* Metrics serve as a basis of cost estimating, training planning, and resource planning, scheduling, and budgeting.

(*ii*) *Organizing:* Size and schedule metrics influence a project's organization.

(*iii*) *Controlling:* Metrics are used to status and track software development activities for compliance to plans.

(*iv*) *Improving:* Metrics are used as a tool for process improvement and to identify where improvement efforts should be concentrated and measure the effects of process improvement efforts.

6.1 Importance of Metrics

Deriving metrics in every phase of the SDLC has a major importance through out the life cycle of Software for management, Managers, Developers and Customers [17].

6.1.1 Metrics in Project Management

1. Metrics make the project's status visible. Managers can measure progress to discover if a project is on schedule or not.
2. Metrics focus on activity. Workers respond to objectives, and metrics provide a direct objective for improvement.
3. Metrics help to set realistic expectations. By assisting in estimation of the time and resources required for a project, metrics help managers set achievable targets.
4. Metrics lay the foundations for long-term improvement.
5. Metrics also help the management in reducing various resources namely people, time and cost in every phase of SDLC.

6.1.2 Metrics in Decision Making

Metrics will not drive a return on investment unless managers use them for decision making. Some decisions in which software metrics can play a role include Product readiness to ship/deploy, Cost and schedule for a custom project, How much contingency to include in cost and schedule estimates, Where to invest for the biggest payback in process improvement, and When to begin user training.

Managers should demand supporting metrics data before making decisions such as these. For example, they can use fault-arrival-and-close-rate data when deciding readiness to deploy. Knowing the Overall project risk through metrics can help managers decide how much contingency to include in cost and schedule estimates.

6.2 Procedural (Traditional) Software Metrics

Support decision making by management and enhance return on the IT investment [18]. Once business goals have been identified, the next step is to select metrics that support them. Various types of Metrics [20] found in literature are:

This document was created using SOLID CONVERTER PDF To remove this message, purchase the product at www.SolidDocuments.com

Lines of Code namely Total Lines of Count (TLOC), Executable Lines of Count (ELOC), Comment Lines of Count (CLOC).

Hallstead's Metrics namely Program Length (N), Program Volume (V), Effort to Implement (E), Time to Implement (T), Number of Delivered Bugs (B).

Function point (FP) [20] is a metric that may be applied independent of a specific programming language, in fact, it can be determined in the design stage prior to the commencement of writing the program.

6.3 Object Oriented Metrics

The most commonly cited software metrics to be computed for software with an object-oriented design are those proposed by Chidamber and Kemerer [20], [21]. Their suite of metrics consists of the following metrics: weighted methods per class, depth of inheritance tree, number of children, coupling between object classes, response for a class, and lack of cohesion in methods.

6.4 Coverage Metrics

To measure how well the program is exercised by a test suite, coverage metrics are used [21, 22]. There exists a number of coverage metrics in literature. Following are descriptions of some types of coverage metrics.

(*a*) *Statement Coverage:* This metric is defined as "The percentage of executable statements in a component that have been exercised by a test case suite".

(*b*) *Branch Coverage:* This metric is defined as "The percentage of branches in a component that have been exercised by a test suite".

(*c*) *Loop Coverage:* This metric is defined as "The percentage of loops in a component that have been exercised by a test suite".

(*d*) *Decision Coverage:* This metric is defined as "The percentage of Boolean expressions in a component that have been exercised by a test suite" [22].

(*e*) *Condition Coverage:* This metric is defined as "The percentage of decisions in a component that have been exercised by a test suite". [13].

7. EXPERIMENTAL DATABASES USED FOR COVERAGE METRICS COMPUTATION

For experimental purpose we have considered four databases created by the under graduate students and two database from UCI Machine Learning Repository [http://archive.ics.uci.edu/ml/].

The details of the above databases are shown in Table 1 as follows. In Table 1, for testing the various databases we have considered 15 test cases for department database and 22 test cases from university dataset, etc. For each and every database, the test cases were executed using the DbUnit tool.

Table 1
Details of the Databases

Database application	*Attributes*	*Records*	*Primary keys*	*Foreign keys*	*Size (KB)*	*Select queries (test cases)*
Census (UCI)	14	682	4	2	62	25
Employee	12	425	5	1	53	30
Department	10	321	3	1	46	15
Location	8	425	2	1	51	20
Tax	11	250	3	1	32	18
University Data Set (UCI)	17	285	7	3	37	22

8. RESULT ANALYSIS

8.1 Analysis of Coverage Metrics

We have calculated the various coverage metrices using coverage tree and command form. We have considered test cases (SELECT queries) which have conditions based on one or more tables as shown in Table 2.

From Table 2 it is clear that as the number of tables and the conditions in the SELECT queries increases, the coverage percentage for both metrices using coverage tree and command form gets decreased.

For example considering test Case 4, test Case 5 which contains 1 Table 1 condition, has a coverage tree percentage of 49.61%, 45.66% and command form percentage of 82.56%, 79.45% respectively. For test case 23,24 which has 2 Table 2 conditions, the percentages is reduced to 21.69%, 24.87% and 68.22%, 69.63% respectively. Similarly considering test case 35, 36 that has 4 Table 2 conditions the percentage further reduces to 25.00%, 17.94% and 66.45%, 67.41% respectively. Similarly considering test case 43 that has 5 Tables 4 conditions the percentage further reduces to 20.12% and 66.34% respectively.

The results of two Coverage algorithms namely coverage tree and command form were analyzed for different kinds of select statements with varying tables

This document was created using SOLID CONVERTER PDF To remove this message, purchase the product at www.SolidDocuments.com

Table 2
Calculation of the Various Coverage Metrices

Test case no.	*Query*	*Table*	*Condition*	*Coverage % by coverage tree*	*Coverage % by command form*
1.	Select * from census where IDNO=11;	1	1	50.00	83.33
2.	Select * from census where name='Prameela';	1	1	46.72	80.41
3.	Select * from census where job='manager';	1	1	47.32	81.21
4.	Select * from census where salary=20000;	1	1	49.61	82.56
5.	Select * from census where sex='F';	1	1	45.66	79.45
6.	Select * from emp, dept where emp.eid=121;	2	1	33.3	75.12
7.	Select * from emp, dept where emp.ename='Prameela';	2	1	35.24	69.23
8.	Select * from emp, dept where emp.sal=35000;	2	1	32.87	73.82
9.	Select * from emp, dept where emp.job='SPgmer';	2	1	34.71	70.64
10.	Select * from emp, dept where emp.emgid=191;	2	1	31.87	69.23
11.	Select * from emp, dept where emp.eid=dept.eid and emp.ename='Balaji';	2	2	25.31	71.42
12.	Select * from emp, dept where emp.eid=dept.eid and emp.emgid=191;	2	2	23.76	73.65
13.	Select * from emp, dept where emp.eid=dept.eid and emp.job='SPgmer';	2	2	22.64	75.22
14.	Select * from emp, dept where emp.eid=dept.eid and emp.sal=35000;	2	2	24.87	70.43
15.	Select * from emp, dept where emp.eid=dept.eid and dept.dname='manager';	2	2	21.69	74.77
16.	Select * from emp, dept, loc where emp.eid=121;	3	1	50.00	81.21
17.	Select * from emp, dept, loc where emp.ename='Balaji';	3	1	47.32	80.41
18.	Select * from emp, dept, loc where emp.job='SPgmer';	3	1	49.61	83.33
19.	Select * from emp, dept, loc where emp.sal=35000;	3	1	45.66	82.56
20.	Select * from emp, dept, loc where dept.dname='manager';	3	1	46.72	81.21
21.	Select * from emp, dept, loc where emp.eid=dept.eid and Dept.did=loc.did;	3	2	25.31	74.12
22.	Select * from emp, dept, loc where emp.eid=dept.eid and loc.city='Banglore';	3	2	23.76	70.65
23.	Select * from emp, dept, loc where emp.eid=dept.eid and dept.did=221;	3	2	21.69	68.22
24.	Select * from emp, dept, loc where emp.eid=dept.eid and emp.sal=35000;	3	2	24.87	69.63
25.	Select * from emp, dept, loc where emp.eid=dept.eid and dept.dname='manager';	3	2	22.64	72.62
26.	Select * from emp join dept on (emp.eid=dept.eid) join loc on(dept.eid=loc.eid) and (dept.did=loc.did);	3	3	37.11	67.56
27.	Select * from emp join dept on (emp.eid=dept.eid) join loc on(dept.eid=loc.eid) and (dept.dname='manager');	3	3	25.14	65.87
28.	Select * from emp join dept on (emp.eid=dept.eid) join loc on(dept.eid=loc.eid) and (emp.sal=35000);	3	3	17.98	63.91
29.	Select * from emp join dept on (emp.eid=dept.eid) join loc on(dept.eid=loc.eid) and (loc.city='Banglore');	3	3	32.33	59.65
30.	Select * from emp join dept on (emp.eid=dept.eid) join loc on(dept.eid=loc.eid) and (emp.job='SPgmer');	3	3	16.77	61.56
31.	Select * from emp, dept, loc, mng where emp.eid=dept.eid;	4	1	49.56	69.23
32.	Select * from emp, dept, loc, mng where dept.did=loc.did;	4	1	47.77	68.22

Table Contd.

This document was created using SOLID CONVERTER PDF To remove this message, purchase the product at www.SolidDocuments.com

Table 2 Contd.

33.	Select * from emp, dept, loc, mng where dept.mid=mng.mid;	4	1	50.00	70.65
34.	Select * from emp, dept, loc, mng where emp.eid=dept.eid and dept.mid=mng.mid;	4	2	23.65	68.12
35.	Select * from emp, dept, loc, mng where emp.eid=dept.eid and dept.did=loc.did;	4	2	25.00	66.45
36.	Select * from emp, dept, loc, mng where emp.eid=dept.eid and dept.did=loc.did and dept.mid=mng.mid;	4	3	17.94	67.41
37.	Select * from emp, dept, loc, mng where emp.eid=dept.eid and dept.did=loc.did and loc.city='Banglore';	4	3	19.44	66.56
38.	Select * from emp join dept on (emp.eid=dept.eid) join loc on(dept.eid=loc.eid) join mng (dept.mid=mng.mid) and (mng.mname='Narayanan');	4	4	15.25	67.34
39.	Select * from emp join dept on (emp.eid=dept.eid) join loc on(dept.eid=loc.eid) join mng (dept.mid=mng.mid) and (mng.did=122);	4	4	12.51	67.56
40.	Select * from emp, dept, loc, mng, tax where dept.mid=mng.mid;	5	1	49.56	75.34
41.	Select * from emp, dept, loc, mng, tax where emp.eid=dept.eid and dept.mid=mng.mid;	5	2	25.12	69.89
42.	Select * from emp, dept, loc, mng, tax where emp.eid=dept.eid and dept.eid=loc.eid dept.mid=mng.mid;	5	3	22.76	68.67
43.	Select * from emp join dept on (emp.eid=dept.eid) join loc on(dept.eid=loc.eid) join mng (dept.mid=mng.mid) join tax (emp.eid=tax.eid);	5	4	20.12	66.34
44.	Select * from emp join dept on (emp.eid=dept.eid) join loc on(dept.eid=loc.eid) join mng (dept.mid=mng.mid) join tax (emp.eid=tax.eid) and (mng.mname='Narayanan') ;	5	5	51.00	67.22

and conditions as shown in Table 2 and the graphical representation of the result is as shown in Fig. 6. From the table it is clear that the coverage percentage calculated using command form is higher than the coverage tree algorithm. Thus these metrices help the test manager to keep track of how much testing they have performed for the database application.

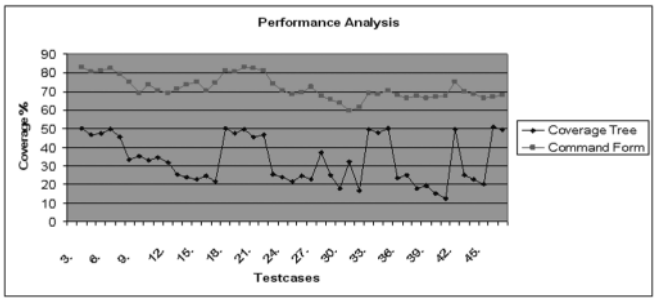

Fig. 6: Performance Analysis of two Coverage Algorithms

8.2 Analysis of Execution Time for Database Application

In this paper we have compared the execution time of the complete Database Testing without agents and with agents. The result is shown in Table 3.

From Table 3 it is clear that the execution time reduces from 29.86 seconds to 10.678 seconds for the census application with 682 records with 15 test cases. Similarly for employee and university dataset applications there is a reduction in execution time when multi-agents are used. Thus MADT framework helps the test manager in performing the automated testing in an efficient, cost effective and time consuming manner.

Table 3
Analysis of Execution Time for Database Application

Database application	*No. of test cases*	*Execution time using DbUnit (sec)*	*Execution time using JADE agents (sec)*
Census	15	29.86	10.678
Employee	15	36.91	15.954
University Data Set	15	21.71	10.865

9. CONCLUSION AND FUTURE ENHANCEMENT

Managing the test process, being a complex and time consuming, has been drastically reduced through automation using multi-agents thereby increasing the Quality of the Test management. Well-designed metrics

This document was created using SOLID CONVERTER PDF To remove this message, purchase the product at www.SolidDocuments.com

with documented objectives can help an organization obtain the information it needs to continue and improve its software products, processes, and services. We have proposed a test management framework that eases the job of the Test Manager for a database application using multi-agents.

In this paper we have evaluated the two coverage algorithms using coverage tree and command form for database applications to prove the effectiveness of testing.

This work can be extended by calculating the coverage for other DML statements of database applications. In order to attain the better accuracy and effectiveness of testing, SQL mutation operators could be used for SQL queries. Using mutation operators the average percentage of fault detected, fault defect density of a particular test case can be calculated.

REFERENCES

[1] M.Y. Chan and S.C. Cheung, "Testing Database Applications with SQL Semantics", *In the Proceedings of 2nd International Symposium on Cooperative Database Systems for Advanced Applications (codas'99)*, Wollongong, Australia, March 1999, pp. 363–374.

[2] D. Chays, P. Frankl, et. al., "A Framework for Testing Database Application", *ACM International Symposium on Software Testing and Analysis*, Portland, Oregon, 2000.

[3] Edward Hung, "Testing of Database Applications", the University of Maryland, College Park. 2001.

[4] David Chays, Yuetang Deng. "Demonstration of AGENDA Tool Set for Testing Relational Database Applications". *Proceedings of the 25th International Conference on Software Engineering (ICSE'03)*, 2003. *IEEE*.

[5] Yuetang Deng Phyllis Frankl Zhongqiang Chen. "Testing Database Transaction Concurrency". *Proceedings of the 18th IEEE International Conference on Automated Software Engineering (ASE'03)*, 2003, *IEEE*.

[6] Xintao Wu, Chintan Sanghvi, Yongge Wang, Yuliang Zheng. "Privacy Preserving Database Application Testing". WPES'03, October 30, 2003, Washington, DC, USA. 2003, ACM.

[7] Ramkrishna Chatterjee, Gopalan Arun, Sanjay Agarwal, Ben Speckhard, and Ramesh Vasudevan. "Using Applications of Data Versioning in Database Application Development". *Proceedings of the 26th International Conference on Software Engineering (ICSE'04)*, 2004. *IEEE*.

[8] S.K. Gardikiotis, N. Malevris, T. Konstantinou. "A Structural Approach Towards the Maintenance of Database Applications". *Proceedings of the International Database Engineering and Applications Symposium (IDEAS'04)*, 2004, *IEEE*.

[9] Yuetang Deng, Phyllis Frankl, David Chays. "Testing Database Transactions with AGENDA. ICSE'05, May 15–21, 2005, St. Louis, Missouri, USA. 2005 ACM.

[10] W.K. Chan, S.C. Cheung and T.H. Tse, "Fault-Based Testing of Database Application Programs with Conceptual Data Model", *to appear in the Proceedings of the 5th International Conference on Quality Software*, 2005. IEEE Computer SocietyPress.

[11] www.dbunit.org.

[12] Gregory M. Kapfhammer, Mary Lou Soffa. "A Family of Test Adequacy Criteria for Database-Driven Applications". ESEC/FSE'03, September 1–5, 2003, Helsinki, Finland, 2003, ACM.

[13] Howden, "Weak Mutation Testing and Completeness of Test Sets", *IEEE Trans. Software Eng.*, **SE-8 (4)**, July 1982, pp. 371–379.

[14] William G.J. Halfond and Alessandro Orso. "Command-form Coverage for Testing Database Applications". *21st IEEE International Conference on Automated Software Engineering (ASE'06)*, 2006, *IEEE*.

[15] María José, Suárez-Cabal and Javier Tuya, "Using an SQL Coverage Measurement for Testing Database Applications", *ACM SIGSOFT Software Engineering Notes*, **29 (6)**, pp. 253–262, November 2004.

[16] Norman E Fenton, Martin Neil, "Software Metrics: Roadmap".

[17] Qaiser Durrani, "Role of Software Metrics in Software Engineering and Requirements Analysis", 2005, *IEEE*.

[18] Grady R.B, "Practical Software Metrics for Project Management and Process Improvement", Prentice-Hall, 1992.

[19] Roger S. Pressman, "Software Engineering: A Practitioner's Approach".

[20] Dekkers C., "Demystifying Function Points: Let's Understand Some Terminology". *IT Metrics Strategies*, Oct. 1998.

[21] Chidamber S.R. and Kemerer R.F., "A Metrics Suite for Object-oriented Design". *IEEE Trans. Software Eng.*, **20 (6)**, (June 1994), 476–493.

[22] John Joseph Chilenski and Steven P. Miller, "Applicability of Modified Condition/Decision Coverage to Software Testing", *Software Engineering Journal*, September 1994, **9 (5)**, pp. 193–2000.

[23] Roberta Coelho, Elder Cirilol, Uirai Kuleszal, Arndt von Staa, Awais Ashid, Carlos Lucenal, "JAT: A Test Automation Framework for Multi-Agent Systems", *ICSM072007 IEEE*.

[24] Yuji Kosuga, Kenji Kono, Miyuki Hanaoka, iho Hishiyama, Yu Takahama, "Sania: Syntactic and Semantic Analysis for Automated Testing against SQL Injection", *23rd Annual Computer Security Applications Conference 2007 IEEE*.

[26] Zunliang Yin, Chunyan Miao, Yuan Miao and Zhiqi Shen, "Actionable Knowledge Model for GUI Regression Testing", *IEEE*, Melbourne, Australia, 2005, pp. 165–168.

[27] Chia-En Lin, Krishna M. Kavi, Frederick T. Sheldon, Kris M. Daley and Robert K. Abercrombie, "A Methodology to Evaluate Agent Oriented Software Engineering Techniques", *IEEE*, Waikoloa, HI, 2007, pp. 60.

This document was created using SOLID CONVERTER PDF To remove this message, purchase the product at www.SolidDocuments.com

[28] Xiaoying Bai, Guilan Dai, Dezheng Zu and Wei Te Tsai, "A Multi-Agent Based Framework for Collaborative Testing on Web Services", *Proceedings of the Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems and Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA'06)*, 2006. *IEEE*.

[29] Yu Qi; Kung D.; Wong E., "An Agent-Based Testing Approach for Web Applications", *Computer Software and Applications Conference*, 2005. COMPSAC 2005. *29th Annual International*, **2**, 26-28, July 2005, pp. 45 – 50.

[30] www.jade.tillab.com.

This document was created using SOLID CONVERTER PDF To remove this message, purchase the product at www.SolidDocuments.com