Advanced Science and Technology Letters
Vol.45 (AST 2014), pp.86-89
http://dx.doi.org/10.14257/astl.2014.45.17

Testing Algebra for Data-driven Testing

Jae-Han Cho[1] and Lee-Sub Lee[2] *

Department of Computer Engineering, Kumoh National Institue of Technology University,
61, Daehak-ro, Gumi-si, Gyeongsangbuk-do, 730-701 South Korea
{ [1]jaehanfs@gmail.com, [2]eesub@kumoh.ac.kr }

Abstract. Data-driven testing was introduced in order to reduce the amount of work of generating test cases. The method was proposed to solve the dependency Record-and-playback. Testers could focus on the generating test data so that it contributes to the improvement of productivity. Since a lot of testing data is required, the workload of testers is explosively increasing. In this paper, for easy generation of test data, we propose a testing algebra. Testing algebra consists of operators for generating test data. Automatic generation of test data from a described algebraic expression is also proposed. With the testing algebra, testers can design, implement and manage test data for Data-driven testing efficiently.

Keywords: Software Testing, Relational algebra, Data-driven testing, Testing algebra, Test driver.

1 Introduction

In order to develop high-quality software, testing has been more and more important recently. The purpose of testing is improving the quality of software by minimizing the faults of software systems. The acceptance test is the last resort of ensuring the high-quality software. With a huge number of trivial tasks of writing test cases, well-trained testing experts should be required. Thus the recent trend is separating testers from developers. The black-box testing techniques are generally applied to the acceptance test.

Nowadays several automatic testing frameworks are introduced for improving Record-and-play back, such as Data-driven and Action-driven, etc. In this paper, we proposed the testing algebra to generate test data easily. This paper focused on Data-driven testing, because it is a basis of keyword-driven testing, has simple structure, and is easy to reduce the large number of manually writing test cases.

2 Testing Algebra for Data-Driven Testing

In the Record-and-playback if testers want to improve test data for continuous regression testing test script should be rewritten also. Data-driven testing by separating test data from test scripts, testers can focus on writing test data without

ISSN: 2287-1233 ASTL

concerning about specific testing tools. Eventually the writing test data is more important than the writing data script. The Data-driven testing is described in the following Figure 1.

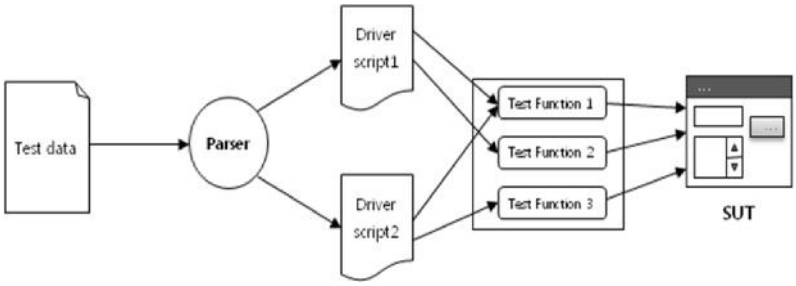

Fig. 1. Data-driven testing.\

Since the number of test cases is very huge in the Data-driven testing, it is difficult to maintain the script for Quality Assurance (QA) experts. In this paper, we proposed a method of generating the test data from test algebraic expressions for performing Data-driven Testing. The following Figure 2 shows of overview of the proposed method.

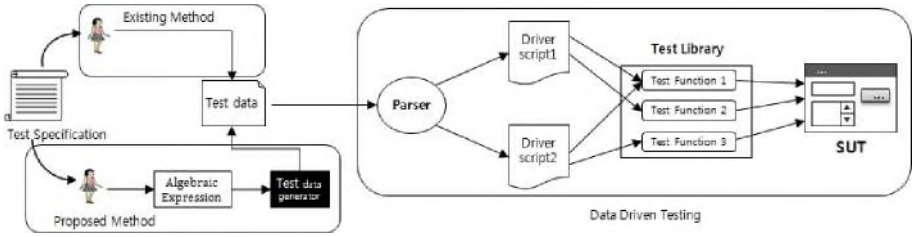

Fig. 2. Generating test data from a test algebraic expression.

The testing data generator has been implemented as followings:
- Language: Python 3.x.
- Tool: Python IDE and Eclipse.
- Data Sheet: CSV of Excel.

The class diagram for implementation of the proposed method was shown in Figure 3. The Field class is defined for dealing with primitive test data. It contains various automatic test data creation operators such as boundary partition, enumeration, and regular expression. The Table class is defined for composing test data for Data-driven testing. It utilizes Field objects as building blocks. It contains various operators for managing relationships among Field objects. Main, TableList, and FieldList are trivial helper classes for managing Field and Table objects.

Fig. 3. Testing algebraic Implementation Model.

3 Case Study

An example of a calculator program is shown in Figure 4. A corresponding test data was written using testing algebra.

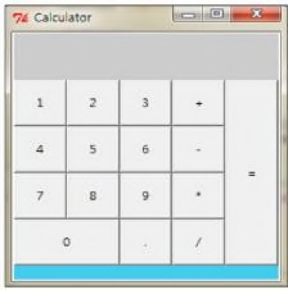

Fig. 4. An example of a calculator program.

If a tester manually writes test cases then he should write and manage 324 lines. By using the testing algebra, it is possible to express less than 20 lines. An example of an expression is follows:

```
table = Table(“calculator Test”)
op1 = gBoundary(“operand1”, intMax, intMin, delta)
op2 = gBoundary(“operand2”, intMax, intMin, delta)
```



```
op = gField(“operator”, [‘+’, ‘-’, ‘*’, ‘/’])

table.addField(op1)
table.combination(op)
table.combination(op2)

table.createEmptyField(“oralceData”)
table.appendExistingField(“oracleData”, ”operand1”)
table.appendConstant(“oracleData”, “ ”)
table.appendExistingField(“oracleData”, ”operator”);
table.appendConstant(“oracleData”, “ ”)
table.appendExistingField(“oracleData”, ”operand2”)

table.createEmptyField(“oracle”)
table.evaluateField(“oracle”, “oracleData”)

table.storeCSV(“data.csv”)
testDriver(“data.csv”)
```

4 Conclusion

The research proposed a method of the expressing by testing algebra and automatic generating of the testing data with it. The testing data could be built more convenient way and so offered more efficient modification and management for better quality software. The future work will include an efficient formally describing method for Keyword/Action-driven testing.

References

1. Yu Seung Ma, Hui Sui Seo, Hyun Seop Bae, Yong Rae Kwon.: A Constraints-based Testing Environment for Non-Sequential Software. Journal of Computing Science and Engineering(JCSE), 611-613(1999)
2. Poore, J.H, Trammell, C.J.: Application of statistical science to testing and evaluating software intensive systems. In: Science and Engineering for Software Development: A Recognition of Harlan D. Mills' Legacy, 1999. Proceedings, pp. 40--57. IEEE Press, Los Angeles, CA (1999)
3. Jong-Hak Lee .: A Tuning Algorithm for the Multidimensional Type Inheritance Index of XML Databases. Journal of Korea Multimedia Society Vol. 14. No. 2, pp.269-281(2011)
4. Hoyoung Jeong, Jungmin Kim, Junwon Jung, Jongnam Kim, Donghyuk Im, Hyoung-Joo Kim.: RDF and OWL Storage and Query Processing based on Relational Database. Journal of Computing Science and Engineering (JCSE), 451-457(2005)
5. Sang-Woo Maeng, Hong-Seong Park.: Comparing Black-box Testing Methodology and Performance Measurement by Test Coverage Analysis. The Korean Institute of Electrical Engineers, 14-17(2009)

