Research on Lottery Game Automation Test Framework

Zhengjun Pan [a)], Jian Hee [b)]

Department of Informatics, Beijing University of Technology, Beijing 100021, China

[a)] Corresponding author: pzjcjy@gmail.com
[b)] jianhee@bjut.edu.cn

Abstract. The main research content of this topic is the automatic testing framework of lottery games. The research and design of the two test frameworks, data-driven and game-rule-driven, respectively, analyzes and studies the editing, storage and processing of test data. At the same time, the testing framework based on lottery game rules will also focus on multi-level keyword technology.

Key words: automatic testing framework; data-driven; game-rule-driven; multi-level keyword.

DATA DRIVEN LOTTERY GAME TESTING RESEARCH

In simple test scripts, test data is embedded. This leads to the problem that once the test data is updated or changed, the test script code is bound to have changed. This may not be a problem for developers who implement test script code, but it is relatively difficult for test engineers who do not have much experience. And if the script's code is long and unstructured, this is a difficult task for anyone [1]. Another problem with embedding test data in test scripts is that for many similar tests, there may be similar test data, but if once the entire project has a specific data to update, all test scripts must be performed. Update. Therefore, when faced with establishing a large-scale automated testing framework, embedding data in test scripts is obviously not a feasible solution [2]. Therefore, a more ideal solution is to read the data required for the test from an external data source and execute the test according to it. This solution is called data-driven testing, as shown in the Fig.1

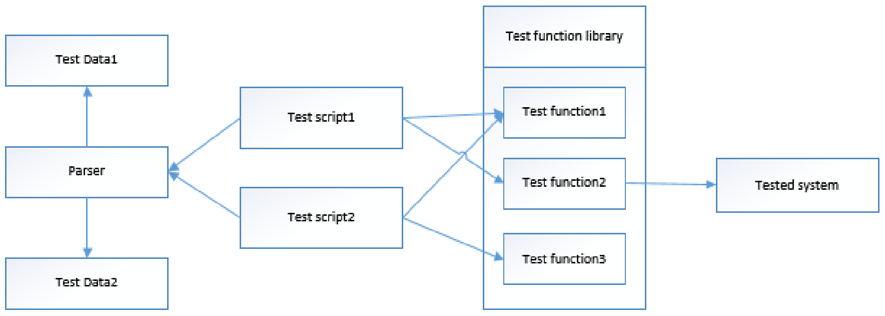
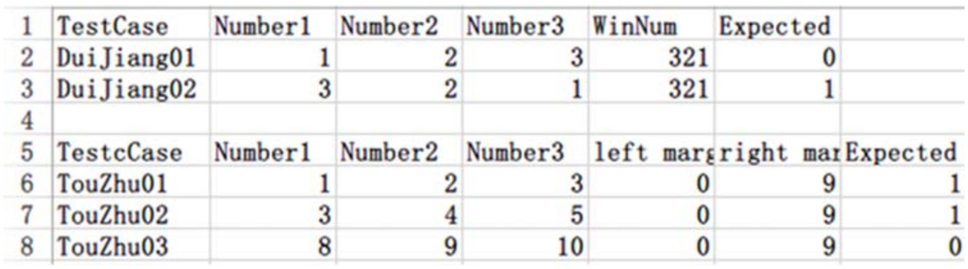


FIGURE 1. Data driven framework.

For test engineers without any programming experience, external data sources must be easily and easily changed. General data sources are presented in tabular form and are changed through a spreadsheet program. As shown Fig.2



1	TestCase	Number1	Number2	Number3	WinNum	Expected	
2	DuiJiang01	1	2	3	321	0	
3	DuiJiang02	3	2	1	321	1	
4							
5	TestcCase	Number1	Number2	Number3	left marg	right mar	Expected
6	TouZhu01	1	2	3	0	9	1
7	TouZhu02	3	4	5	0	9	1
8	TouZhu03	8	9	10	0	9	0

FIGURE 2. External test data example

Edit and Store Test Data

Because data-driven test data comes in tabular form, it is stored and edited using a spreadsheet program. Both the test engineer and the business person understand the spreadsheet program and can operate it [3]. Another benefit is that the spreadsheet program files are very easy to manage, making it easy to move or copy files that store the same data.

Processing Test Data

For a modern scripting language, implementing a script that parses data driven test data is very simple. That is, the data in the above figure can be parsed and imported using a four-line Python code

```
data=open('testdata.tsv'). read ()
lines=data. splitlines()[1:] #[1:]excludes the header row
for line in ines:
testId, number1, number2, number3, winNum,
expected=line.split('\t')
#Actual test functionality excluded
```

In the above code, the test data is first read as a data variable and then separated line by line to exclude the header line. The data line is assigned to the specified variable by item, and the code in the script divides the table into a single cell and assigns the data to each variable so that each test data is available.

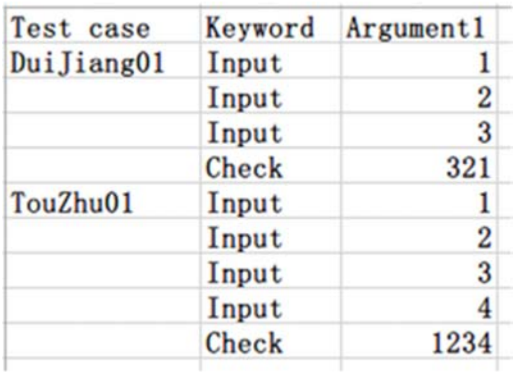
The entire parsing process is in a separate parsing script, which is also inconsistent with the previously mentioned and promoted modularity. Moreover, this parser has its own functional limitations. That is, if empty rows or empty columns appear in the data, parsing will fail [4]. Therefore, more parsing functions should be implemented as a common function in the library, which can be called by all driver scripts. Further implementation of related functions will be discussed in the following sections.

RULE-DRIVEN LOTTERY GAME TEST RESEARCH

The previous section briefly introduced the test method driven by lottery game data and summarized it. One of the biggest limitations mentioned is that all test cases need to be similar. Once a new type of test case is written, the test engineer needs to have programming capabilities. In order to deal with this kind of problem, need another kind of method - based on the keyword to drive the test method, namely the lottery game rule drives the test. At this time, the external input data not only needs to store the test data, but also needs some instructions on how to handle the data [5]. This is the information separated from the test script. These instructions are called keywords, and test engineers have the flexibility to use these keywords to build test cases. The remaining basic methods - reading data from external files and performing tests based on this - these are the same as data driven. Logically speaking, keyword-driven is an extension of data-driven.

Edit and Store Test Data

There is no essential difference between this part and the data drive. Either an electronic data table or an HTML form can be applied to this kind of test, but if it is a very large-scale project, the use of database storage can solve the problem of data expansion. As shown Fig.3



Test case	Keyword	Argument1
DuiJiang01	Input	1
	Input	2
	Input	3
	Check	321
TouZhu01	Input	1
	Input	2
	Input	3
	Input	4
	Check	1234

FIGURE 3. Keyword external test data example

Processing test data

The simpler keyword-driven test data shown in the figure above is processed and processed in much the same way as data-driven test data. Same as the code shown in the previous chapter, get the keyword, and get the parameters for the driver script. In a more complex example, the parser should be implemented as a universally available module in the system.

When the test data is parsed, the driver script needs to be able to "translate" the keywords and then perform specific actions based on the parameters therein. Each keyword corresponds to a processing program, and the processing program is all placed in the function library, which can effectively ensure the modularity of the entire framework. As shown Fig.4.

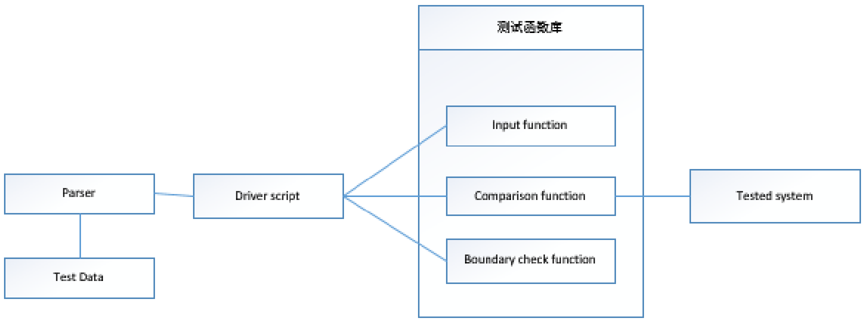


FIGURE 4. Drive script processing data function diagram

Multilevel Keyword Technology

Hierarchical Keyword Separation Test can be designed according to the first, second and third levels, and its functions can be divided into test cases, test case business logics and test steps. Automated test systems are the result of layer-by-layer invocation of use cases, enabling the system's use cases to be refined. The test table of the first-level keywords is an outline design, which lists the various function points organically, so that the system's test function is more comprehensive, involves a wider range of aspects, and the test case files are more efficient [6]. In addition, high-level keyword pairs can be driven by index search secondary keywords. The second-level keyword-driven test is a detailed test of the system. Each use case of the system can be logically tested. The second-level keyword can search the third-level keyword information search table. The third-level keyword test is a detailed step, which tests the smallest unit, and the second-level keyword-driven test and the third-level keyword-driven test are combined test suites.

Hierarchical Keyword Separation Test can be designed according to the first, second and third levels, and its functions can be divided into test cases, test case business logics and test steps. Automated test systems are the result of layer-by-layer invocation of use cases, enabling the system's use cases to be refined. The test table of the first-level keywords is an outline design, which lists the various function points organically, so that the system's test function is more comprehensive, involves a wider range of aspects, and the test case files are more efficient. In addition, high-level keyword pairs can be driven by index search secondary keywords. The second-level keyword-driven test is a detailed test of the system. Each use case of the system can be logically tested. The second-level keyword can search the third-level keyword information search table. The third-level keyword test is a detailed step, which tests the smallest unit, and the second-level keyword-driven test and the third-level keyword-driven test are combined test suites.

ACKNOWLEDGMENTS

I sincerely thank my teacher Jian Hee. Since I just entered my school advisor, I have communicated with me to formulate my three-year learning direction and goals. With the correct goal, we have made a solid step forward. Special thanks to the mentor for trusting me. When I first started school, I had a limited level of programming. The tutors gave me the new project with confidence. Before the project was conducted, I was instructed to conduct targeted learning and personally took me to the library to select books. With the previous theoretical guidance and hands-on project practice, my programming progressed quickly. Not only learned about the needs of the project analysis, coding, testing, deployment and other knowledge and technology, but also learned a lot of teamwork, dealing with customers exchange skills.

REFERENCES

1. Dustin, Elfriede. Effective Software Testing:50 Specific Ways to Improve Your Testing[M]. Boston, Mass: Addison-Wesley,2012.
2. Dustin, Elfriede. Effective Software Testing:50 Specific Ways to Improve Your Testing[M]. Boston, Mass: Addison-Wesley,2012.
3. Troelson, Andrew. C# and the .NET Platform[M]. Berkeley, Calif: Apress, 2011:78-84.
4. GoodEnough J B, Gerhart S L. Toward a Theory of Test Data Selection. IEEE Transactions on Software Engineering, 2010, SE-1(6):156-173.
5. Huang J C. An Approach to Program Testing. ACM Computing Surveys,2010,11-4:113-128.
6. Brannigan V. Acceptance Testing-The Critical Problem in Software Acquisition. IEEE Transactions on Biomedical Engineering, 2010, BME-32(4): 295-299.