Regression Testing for Automatic Fare Collection System of Rail Transit

Mo Cheng[1,a], Maolin Zhang[1,b]

[1]Department of Computer Science and Engineering, Beihang University, Beijing 100191, China

[a]chengmo891@qq.com, [b]zml@buaa.edu.cnl

Keywords: regression testing; automatic fare collection system; cluster analysis; risk assessment; code analysis; functional model

Abstract. Since regression testing technologies are mainly for software systems, there are few research about the access detecting of the auto fare collection system in rail transit. This paper study the multi-level functional modeling, the source code importing of detected object, the historical test information collection, the test of risk assessment of used cases and cluster selection techniques, and integration of those techniques and further study, which aims to put forward a set of complete regression testing method and to implement a suitable regression testing management tool for the access detecting of the automatic fare collection system in rail transit, which is to verify the correctness and feasibility of this method.

Introduction

Basic Concept. The full name of AFC system is the Automatic Fare Collection System [1], and it is a kind of enclosed automatic network system that contains automatic ticketing, checking as well as charging and counting using centralized control by computers, including line central computer (LC), station computer (SC) and station level equipment (SLE). In instance, the Beijing Metro Line Four have a set of LC, 24 sets of SC and 940 units of SLE which consist of 199 TVM, 113 BOM, 510 AG, 48 TCM and 70 PTCM.

Access testing of new line AFC system is the main components of the system, such as connecting LC, SC, SLE, reader, ticket as a whole for consolidated testing after each one fulfilled the standardized testing, and it covers the communication environment testing, certification testing, interface testing, functional testing and interoperability testing.

The frequency of an updated of software is speeding up to satisfy the needs and adaptability of customers' requirements, and it should be supported by regression testing which is to ensure the availability of software that in order of the anticipated patterns. During the whole software testing process, multiple regression testing are required in all stages of software development, in order to stimulating a larger proportion of regression testing workload. Therefore, we have to choose a reduced set of regression test cases for regression testing, because it is impractical to re-run the full set of test cases.

Related Work. In 1993, M.J Harrold et al. proposed test cases intensive technology [2] first time, which makes it possible to effectively reduce the workload of regression testing under the premise of guaranteeing ability of error detection. Demonstrated by this paper, searching for the best test set reduction is a NP-C problem, so generally uses heuristic algorithm to find the approximate solution of problems. Heuristic algorithm mainly includes greedy algorithm [3], HGS algorithm, GE&GRE algorithm and GA algorithm, etc.

In AFC testing center system, regression tests need to be done to detection objects after fixing the problems found in shortlisting detection, prototype detection and access detection, which brings huge regression work. According to study by Akira K. Onoma [4] et al about regression testing application in industrial environment, it must reduce and select test case set in purpose to achieve software process efficiency.

The large-scale software system regression testing reduction technology [5] proposed by A.Orso et al embarks from the program operation difference, which is a test cases selection techniques based on program specification. This method dynamically generates abstract operation. Abstract operation and program formalized specification is consistent on syntax, which is used to describe the actual behavior of program observed in execution of test case set. This technology is mainly divided into two steps. First step, module dividing establishes abstract operation on call relationship between classes and functions in program according to the operation of abstract thinking. Next, statements, methods, classes and modules are divided into changed and unchanged groups according to different granularity, and find elements in unchanged group which establish abstract operation with elements in changed group, and put them into changed group. Second step, selection module regard changed group generated in first step as testing requirements, and take advantage of test requirements coverage techniques [3] to select test cases, ultimately achieve test case intensive reduction.

Solution. In the traditional AFC access testing, we mostly select all test cases for regression testing, or select by professional inspectors based on their previous testing experience. However, the approach not only makes inspectors familiar with AFC testing business, but also can not verify the accuracy and completeness of regression testing, so we need to design a set of regression testing methods according to the characteristics of AFC system and access testing. Based on this, we design two modes for regression testing, regression testing based on functional model, and risk-based regression testing.

The regression testing based on functional model, refers to the study of Beijing local standards, the Rail Toll Collection System Technical Requirements, to sort out detection standards of detection object in access testing, and internal logical structure of detection scheme, then covert them to formal language called the multi-level functional model, and associate them with test cases at last. It can organize and classify test cases effectively. On another hand, it can obtain logical relationship within test cases to fulfill automatic selection of test cases for regression testing. We use source code analysis tool to import and analysis detection object code, and output dependency model, then use model transform tool to generate multi-level functional model.

Risk-based regression testing has two steps, risk assessment and clustering for selection. Risk assessment is the method that estimating every test cases` risk level, and appending them an attribute called risk level for selection in regression testing, and it is automatic. Clustering for selection is the method that using cluster algorithm to classify test cases for identifying them, then can be used for selection as a unit.

AFC System Regression Testing Scenario

General goals of regression testing techniques cover the following points, 1) to increase the error found efficiency of test cases 2) to increase the proportion of code coverage of test cases, 3) testing in an efficient manner without losing credibility, 4) to improve the detection rate of high-risk error, and to decrease the time of high-risk error in entire testing process, 5) to improve the efficiency of error detection caused by modification.

The paper focuses on the point 3 and 4, because the test cases of AFC system access testing are constant, and the access testing needs to be repeated many times in AFC system conventional detection without modification.

Regression Testing Technology Based On Functional Model. Original test cases in AFC system access testing are chaotic, and the relationship between test case and sub test case is not explicit. Based on these problems, we adopt a functional model relied on Beijing local standards, which is divided into models based on Rail Toll Collection System Technical Requirements and used static analysis of source code of detected objects, in order to organizing test cases to test efficiently.

Begin with document. Functional testing is generally based on software requirements, or other software described methods [6], such as Object-Z specifications and UML state diagrams, while in AFC system, we extract the functional features of the Beijing local standards to build the multi-level functional model.

Multi-level functional model is shown below that vertices represent the functional features extract from document, solid lines represent the relationship of points in business process at the same level, and dashed lines represent the detailing of parent point between different levels.

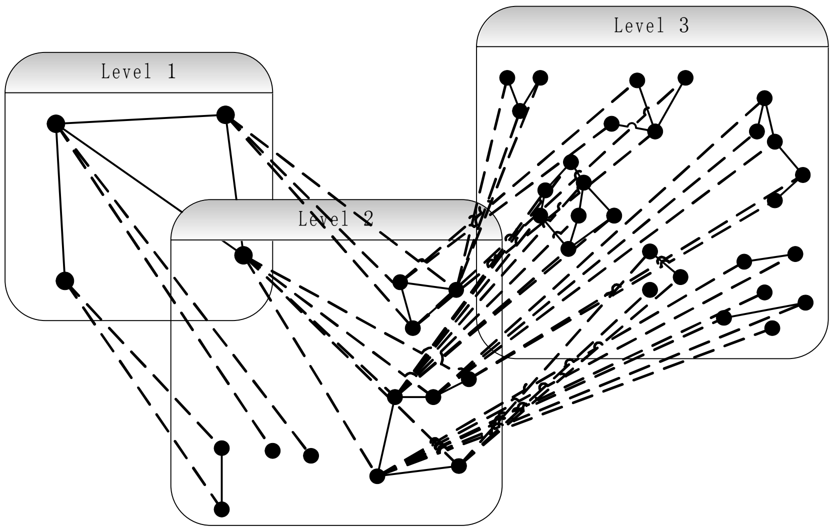


Fig. 1 Schematic diagram of multi-levels functional model

In general, multi-level functional model has around 4 levels, which guarantees it is neither too simple to present model object nor too complicated to application [7]. In accordance with Beijing local model, it has 4 levels. First layer contains function features of line control subsystem, station computer system, ticketing management subsystem, maintenance management subsystem and the equipment function and the key modules function. In instance of parent node of line control subsystem, the layer 2 has nodes of operation management, mode management and so on, and in the layer 3, the nodes related with operation management in layer 2 are operation start, end, time management, etc, and the operation time management node can be divided into run-time schedule management, stations sale timetable, the 24 hours operation and so on.

The relationship between points in different levels (shows in dashed line), can be obtained from Beijing local standards, and in the same levels, business process relationship between nodes can be obtained from manually documents analysis and referring to the business processes defined by Beijing local standards.

The basic idea of the partition and organization of test cases in multi-level functional model [8] is to associate each test case with function feature node, and the logical relationship between test cases can be set up via the logical relationship between same and different levels in multi-level functional model.

The multi-level functional model for each function feature node [9] is defined as below.

```
<class name = “org.apache.tools.ant.util.ReflectUtil” source=
“ant_1.7.0.jar” type = “class” usedBy = “1” usesInternal= “1”
usesExternal= “1” layer = “3”>
  <classRef name= “org.apache.tools.ant.util.ReflectWrapper” type
= “usedBy”/>
  <classRef name= “org.apache.tools.ant.BuildException” type =
“usesInternal”/>
  <classRef name= “java.lang.String” type = “usesExternal”/>
</class>
```

Fig. 2 Definition of point in funcitonnal model

The first line in picture defines the function feature node, including name, sources and so on, and among its structure, the ClassRefs shows the nodes which have connections with this function feature node. And the reason that we use the tag name Class, is to be correspondence with the model build by importing of detection object code. In the classref line, the usedBy attribute represents the nodes from upper layer, and the usesInternal attribute represents nodes from lower layer, usesExternal means from same level. For example, the name of lc.o.otm.s24ho means an address that locate to line control subsystem function, and its sub-layer line is operation management, operation time management, and the 24 hours operation.

Begin with source code of detected objective. The multi-level functional model is not universal because there are a large of detection work about terminal equipment such as AG, TVM, etc, and key module such as ticket box, coin handling module, etc, in AFC system access testing, and if we use test cases supply from manufactures, they may be too particular to execute function module testing using multi-level function model.

The main methods of source code analysis can be divided into static analysis and dynamic analysis. The steps of dynamic analysis is to use source code instrumentation technology that automatically add specific code to the source files under analysis, and then executing a set of test cases to get the actual program running track, finally collect running information [10] for mining the protocols and dependencies of program and so on. On the contrary, static analysis method extracts information from source code directly since we could not use source code instrumentation in detection objection code, and the quality of test cases supply by manufactures cannot been guaranteed, we use the static analysis method to extract the information of source code which include the dependency of classes and packages which been represented by nodes in model, and the dependencies between classes or packages are represented by solid lines.

The definitions of dependency model which build by source code analysis of detection object is similar with multi-level functional model, and are shown below.

```
<class name = "org.apache.tools.ant.util.ReflectUtil" source=
"ant_1.7.0.jar" type = "class" usedBy = "1" usesInternal= "1"
usesExternal = "1" layer = "3">
  <classRef name= "org.apache.tools.ant.util.ReflectWrapper" type
= "usedBy"/>
  <classRef name= "org.apache.tools.ant.BuildException" type =
"usesInternal"/>
  <classRef name= "java.lang.String" type = "usesExternal"/>
</class>
```

Fig. 3 Definition of point in dependency model

The difference between these two models are that the attribute type is added in the class line for distinguish this line is package or class, also for distinguish this relationship belong to package or class. And the attribute usedBy, usesInternal, and usesExternal represent the calling relationship directly, internally and externally.

According to these theories, we develop a prototype of source code static analysis of Java, and export a unit of packages and classes dependency models. Now, the detail steps of the prototype are shown below [11].

1) Import source code document

We use the files have the suffix .class, rather than .java for analysis, and pick up dependency information from the constant pool in the Java virtual machine. In the method area, each type corresponding to a constant pool that stores constants such as string, finally, function name and class name. The constant pool is organized as a list form of entry addresses that can be accessed by index, and the first byte of addresses is a flag represent the type of constant located in it, for example, CONSTANT_Class_info means there are information of a class constant in it.

2) Establish the dependencies of classes

We focus on two constants [12]: the class constant and the UTF-8 constant. The UTF-8 constant stores all character constants including string literal, the full qualified name of class, interface and superclass, class field name and its type name, function name and its return type name, parameter name and type name. And we rely on the class name string constant name, CONSTANT_Class, to distinguish class name and function name in situation that all class names are stored in type of CONSTANT_Utf8.

There are five situations can be defined as directly dependency [13] within objects, 1) superclass, 2) field type, 3) classes of parameters and return value of any method declared or invoked, 4) class implementing an invoked method, 5) string constants if the reflection option is used and the above-mentioned conditions are fulfilled.

Now, we illustrate each situation in code.

```
class MyException extends
Exception{
  int left =
java.awtLable.LEFT;
  MyException(String
msg){
    super(msg);
    Integer n= null;
  }
}
```

```
Class Constant：
MyException
java/lang/
Exception

UTF8 Constant：
(Ljava/lang/
String;)V
```

Fig. 4 Example of inheritance

The referred classes are java.lang.Exception and java.lang.String. It represents Exception and String directly depend on MyException respectively since MyException directly inherited from Exception and String.

```
class ExampleClass{
  int[] counts = new int[5];
  String [][]table = new
String [5][3];
}
```

```
Class Constant：
ExampleClass
java/lang/String
java/lang/Object

UTF8 Constant：
[I
[[Ljava/lang/
String;
()V
```

Fig. 5 Example of object initialization

The referred classes are java.lang.Object and java.lang.String. It represents Object and String directly depend on ExampleClass respectively since ExampleClass includes the initialization of type of int (which is also Object class) and String.

```
class ExampleClass {
  boolean
atLeastOneSystemProperty
=System.getProperties().ke
ys().HasMoreElements();
}
```

```
Class Constant :
ExampleClass
java/lang/Object
java/lang/System
java/util/Hashtable
java/util/
Enumeration

UTF8 Constant:
()V
()Ljava/uti/
Properties;
()Ljava/util/
Enumeration;
()Z
```

Fig. 6 Example of object invocation

The referred classes are java.lang.Object,java.lang.System,java.util.Hashtable, java.util.Properties and java.util.Enumeration. In the ExampleClass, it calls getProperties method to access the properties of class System, secondly, the properties of access is type of Hashtable and access keys() method again. According to above process, the ExampleCLass has a direct dependency relationship with Obeject, System, Hashtable and Enumeration respectively.

```
abstract class
ExampleClass<T extends
Set<Byte>> implements
Collection<Short> {
  T handle(Map<Long, ?
extends Float[]> map){
    return null;
  }
}
```

```
Class Constant :
ExampleClass
java/lang/Object
java/util/Collection

UTF8 Constant :
(Ljava/util/Map;)Ljava/
utilSet;
(Ljava./util/Map<Ljava/
lang/Long;+[Ljava/lang/
Float;>;)TT
<T:: Ljava/util/Set<Ljava/
lang/Byte;>;> Ljava/lang/
Object;Ljava/util/
Collection<Ljava/lang/
Short;>,
```

Fig. 7 Example of function invocation

The referred classes are java.lang.Object, java.util.Collection, java.util.Map, java.util.Set, java.lang.Long, java.lang.Float, java.lang.Byte and java.lang.Short. In the ExampleClass, it defines an invoked method named handle of class T which extends from superclass Set, so as condition 4, class ExampleClass and Set have direct dependency relationship.

3) Establish the dependencies of packages

It is same as the process of establishing dependencies of classes.

4) Import the correspondence with test cases

At last, we should import the correspondence between each function point and test cases.

Risk-Based Regression Testing Technology. In risk-based regression testing, we calculate the error percentage of test cases in history testing and assume that it is the error probability of this test case at next time, and then evaluate the risk cost of test cases by inspectors manually to obtain the risk of each test case. In clustering selection, to classify test cases by risk value into test case groups to be selected in regression testing. It is too simple to characterize test case in risk value, so we also focus the multi-level function model that can be translate into coverage between model and test cases as an attribute of test case that can be clustered to make sure clustering selection is well-rounded. And as shown below, it is the process of risk-based regression testing.

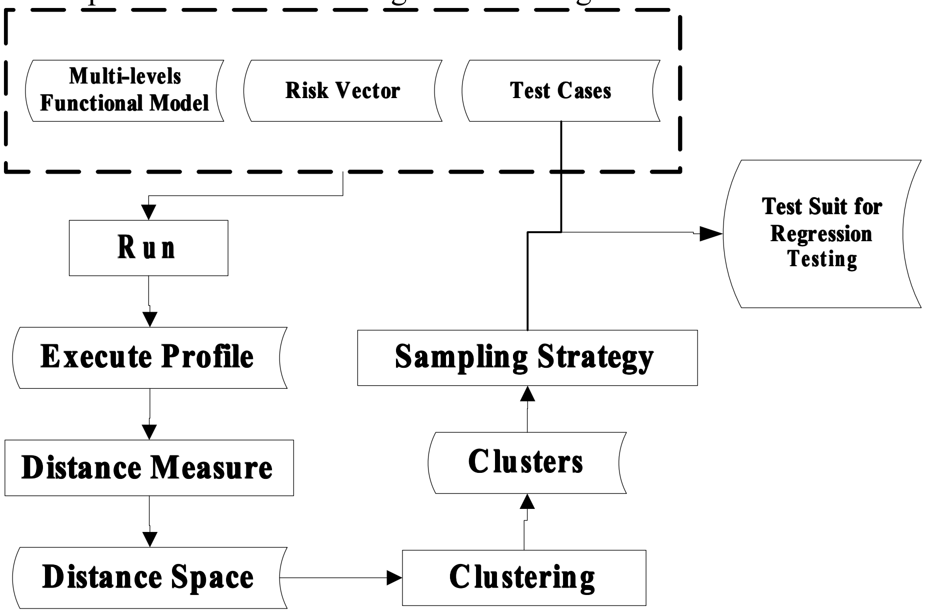


Fig. 8 Process of regression testing based on risk value

Assessment of risk value. The key point of risk-based regression testing technique is to focus on the test case with high risk value which most likely trigger error and can lead to huge cost when it occurs, and ignore test case with low risk value which is irrelevant and with low probability. We give test case two additional attributes, risk cost and risk probability, to assess the cost the error occurs and the probability that test case can find the error, and the result of risk cost and risk probability multiplies is risk value [14].

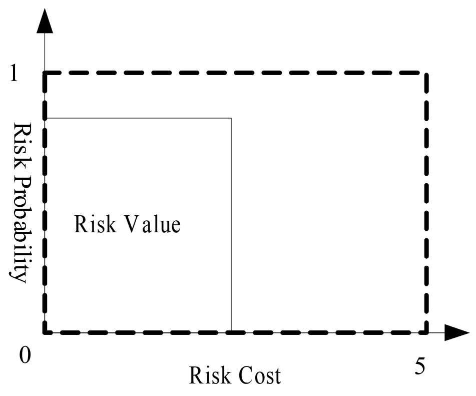


Fig. 9 Definition of risk value

Consider the risk lost in two ways: 1) the cost that users refuse to use the current version of product again when they find an error, 2) the cost that owners repair the current version of product when they find an error [15]. The method of risk assessment is varied such as using multiple linear regression method to reckon the risk probability or using the coverage between errors and test cases to reckon and so on.

Now, we introduce the risk-based regression testing prototype that based on the characteristics of test cases in AFC system access testing.

The risk value is composed of risk cost and risk probability, and its formula is shown below.

(1)

Risk cost can be set by users in range (0, 5), and risk probability is calculated based on history detection results in counting the frequency of risk occurring. Risk probability equals to 1 in default, and variable *t* represents the statistical period.

(2)

Clustering selection. It is not enough to just know the risk value of test cases, so we need to achieve the automatic selection using risk value by clustering selection technique to simplify regression testing, and make sure the accuracy and higher recall rate at the same time.

The basic idea of clustering selection is that test cases in same group have similar behaviors, and the behaviors of test cases in distinct group are quite different.

Clustering selection is generally as follow [16].

1. Collect history execution profile of test cases, such as functions execution profile.
2. Measure the distance between test cases based on risk value, such as Euclidean distance.
3. Use specific clustering algorithm to classify test cases, such as K-means algorithm.
4. Sample from clustering result to constitute regression test cases.

1) Execution profile

In general, the execution profile refers to running information when test cases execute in program, such as function call, module call, etc. And it can be roughly divided into the statement, branch, function, module and others to reflect granularity in different levels. In order to collect historical execution profile, we need to rely on source code instrumentation technology that not fit the AFC system access testing, so we use risk value and multi-levels function model instead.

If clustering is based on risk value, we define test cases as below.

(3)

And, the variable i represents the i-th test case, r_i represents risk value of the i-th test case, n represents the total number of whole test suite.

If clustering is based on multi-levels function model, we define execution profile as below.

(4)

(5)

Among them, the variable i represents the i-th test case, x_i represents execution profile of the i-th test case, x_{ij} represents the relationship between the i-th test case and the j-th test case, and if they have the dependency, x_{ij} =1, on the contrary, x_{ij} =0. The variable m represents the number of functional point in multi-levels function model, and n represents the number of test suite.

2) Distance measure

We use the Euclidean distance [17] to measure the distance between two test cases, and it is also divided into two manners.

If clustering is based on risk value, the distance between x_f and x_g is.

(6)

If clustering is based on multi-levels function model, the distance between x_f and x_g is.

(7)

3) Cluster analysis

Clustering is an unsupervised learning according to attributes of entity to cluster them, in order to maximize the difference between different groups, and to minimize difference in group. The cluster analysis is a kind of similarity analysis which can classify test case that can detect same errors into same cluster. Common clustering algorithms are hierarchical clustering including CHAMELEON, CURE and BRICH, partitioning method including K-means and FREM, and density-based method including OPTICS and DBSCAN [18].

1. Partitioning method. The partitioning method requires pre-established clustering number or cluster center, and then reduce the error of objective function through iterations, when the objective function converges, the cluster result appears.

2. Hierarchical clustering. The other name of hierarchical clustering is tree clustering algorithm that uses the data connection rules in a hierarchical structure, and splits and aggregates the structures repeatedly to gain a cluster result.

3. Density-based method. In the data space, cluster is a dense area which separates from low density area. If the density of dots in an area is greater than the threshold value, we will add this area into near cluster. This method can avoid the weakness of only finding cluster in oval style in distance-based algorithm.

In this paper, a partitioning method to achieve clustering algorithm called K-means algorithm.

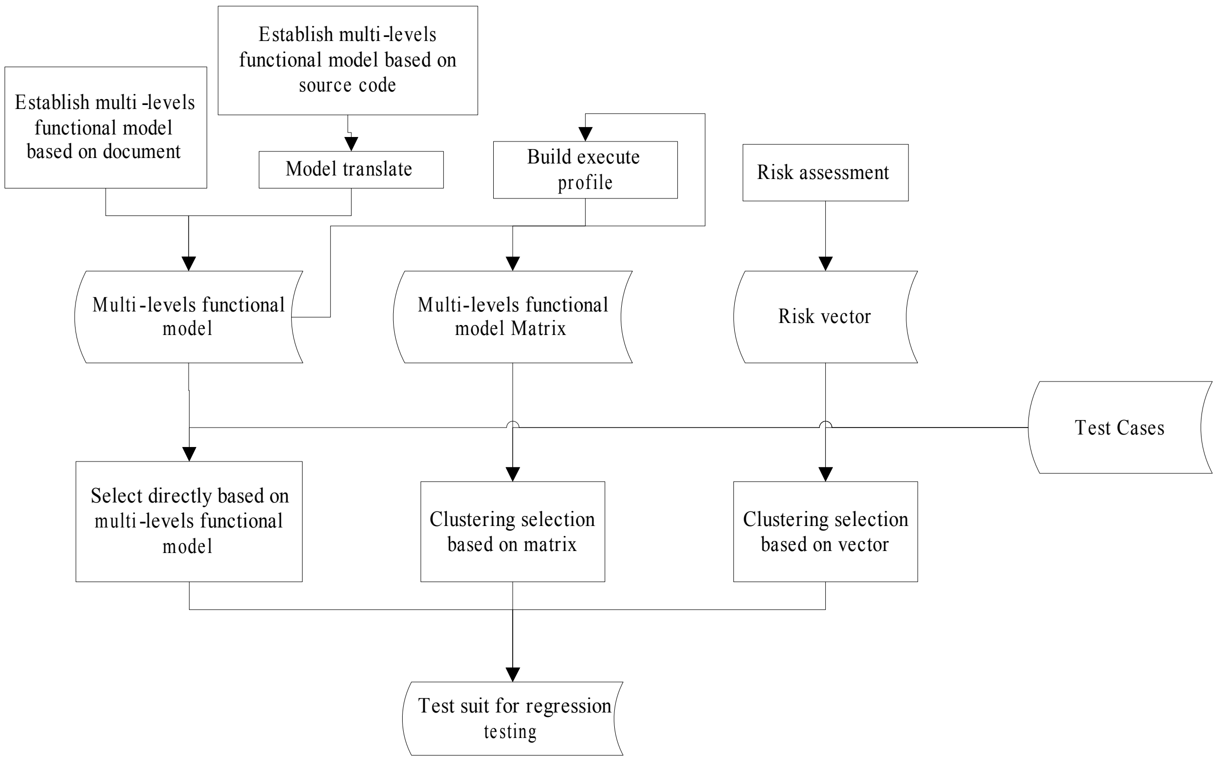
4) Sampling strategy

D. Leon found clustering sampling was more effective than random sampling after assess the effectiveness of sampling strategy, and he divided it into three ways. 1) Sampling in fixed quantity that select *n* samples in each cluster. 2) Sampling in fixed proportion that select *n%* samples in each cluster. 3) Adaptive sampling that select a fixed proportion of samples in each cluster, and if the samples include the test case can detect error, select other test cases into this cluster.

This paper uses two sampling strategies. If clustering is based on risk value, we sample in fixed proportion of 20%. If clustering is based on multi-levels function model, we use adaptive sampling that in 5% firstly. When test cases can detect error, we select all rest of test cases into cluster.

Regression Testing Management Tool of AFC System

Based on the above description of AFC system regression testing technology, we implemented a regression test management tool, and deployed in the Beijing Metro Command Center testing central system. The basic process of tool is shown in Figure.



Establish multi-levels functional model based on source code
Establish multi -levels functional model based on document
Model translate
Build execute profile
Risk assessment
Multi -levels functional model
Multi-levels functional model Matrix
Risk vector
Test Cases
Select directly based on multi -levels functional model
Clustering selection based on matrix
Clustering selection based on vector
Test suit for regression testing

Fig. 10 Process of Regression Testing Management Tool

We install a built-in multi-levels function model bases on Beijing local standard that established by developers manually, and the model can be built by users through import source code of detection objects, and it will set up model automatically by source code static analysis module. And firstly, it will build a dependency model, then translate it to multi-levels function model which has similar structure.

It collects historical test results to count the error times and total test times, to calculate the error probability and assume it is risk probability in the risk assessment module, finally, combine this with risk cost provided by inspectors to gain the risk value.

In the test case selection module, users can select test cases for regression testing through model built by Beijing local standards, or through clustering selection based on risk value or multi-levels function model. In selection module based on multi-levels function model, if selecting a function point to do regression testing, we can select corresponding points in same level which have direct dependency, or select points in lower levels which propagate from this function point using the depth-first traversal algorithm. Users can choose different schemes based on requirement. In selection module based on clustering selection, the steps have mentioned above, that select cluster results in principle of sampling strategies.

The structure of tool is shown below.

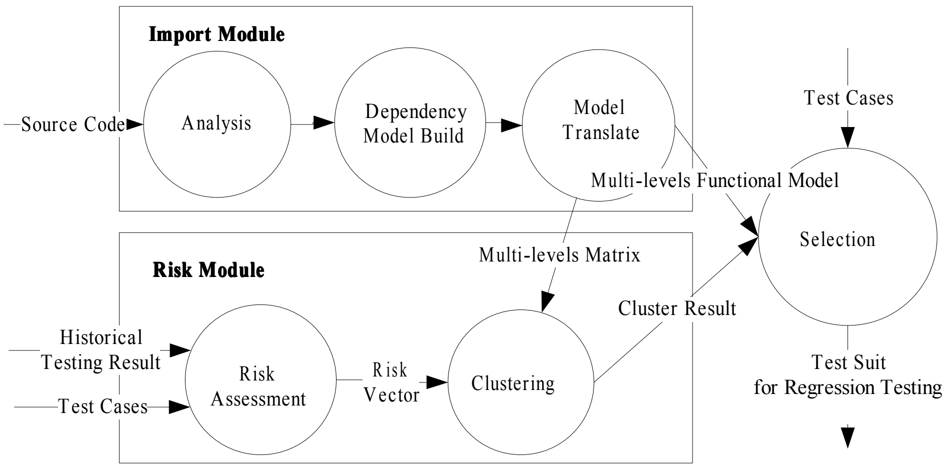


Fig. 11 Framework of Regression Testing Management Tool

Among them, the key modules are divided into source code import module and risk clustering module. Source import module mainly includes source code analysis module, dependency modeling module and model transformation module. Clustering module mainly includes risk assessment module and cluster analysis module. Selection module contains multi-level function model, which refers to selection strategy mechanism that can be selected by users and sampling strategy mechanism in clustering selection.

Summary

The regression testing techniques and completely developed regression testing tool mentioned in the article, are already realized in AFC testing center system of phase 2 of Beijing rail transit traffic command center. Although it is still in the trial period, it has achieved good effect that improves the speed of case selection in regression testing, and guarantee trustworthy and reliability of regression testing.

References

[1] Xue Yun-Lei, Kang Jian-Chu. Research and Implementation of business process management system for AFC Testing Center. Beijing: Beihang University: Computer Application Technology, 2012: 103.

[2] Harrold M J, Gupta R, Soffa M L. A Methodology for Controlling the Size of a Test Suite[J]. ACM Transactions on Software Engineering and Methodology. 1993, 2(3):270-285.

[3] Gu Qing, Tang Bao, Chen Bao-Xu. A Test Suit Reduction Technique for Partial Coverage of Test Requirements. Chinese Journal of Computers. 2001,34.

[4] Akira K. Onoma, Wei-Tek Tsai, Mustafa H. Poonawala, Hiroshi Suganuma. Regression Testing in an Industrial Environment[J]. Communication of the ACM, pages 81-86, volume 41 issue 5, May. 1998.

[5] Alessandro Orso, Nanjuan Shi, Mary Jean Harrold. Scaling Regression Testing to Large Software Systems[J]. ACM SIGSOFT Software Engineering Notes, pages 241-251, volume 29 issue 6, November 2004.

[6] Maharjan S, Dulal N R. A Comparative Study of Component Based Regression Testing Approaches without Source Code[J]. Master Thesis Software Engineering,2011.

[7] Gao Jian-Hua, Chen-Ru. The Research of Cost-Based Testing Technique on the Basis of Multiplayer Functional Model. The 3rd Chinese Testing Conference, 2004.

[8] Jie Kai, Xu Bao-Wen. Component System Regression Testing Model and Technology Research. Nanjing, Southeast University,2006.(in Chinese).

[9] Tsai W T, Bai X, Paul R, et al. Scenario-based functional regression testing[C]//Computer Software and Applications Conference, 2001. COMPSAC 2001. 25th Annual International. IEEE, 2001: 496-501.

[10] Huang Zhou, Peng Xin, Zhao Wen-Yun. Behavior Protocols Recovery Based on Dependency Analysis. Computer Science, 2008,35(9):265.

[11] Jin Jing, Li Meng. Code function recognition approach based on LDA and static analysis. Computer Engineering and Application, 2012.

[12] Huang Ruo-Yi, Mao Cheng-Ying. Test Cases Selection in Regression Testing Based on Dependence Analysis of EFSM. Control & Automation,2005.

[13] Yu Bin, Xu Bao-Wen. Hierarchical and Probabilistic Dependence Analysis of Java Programs. Nanjing: Southeast University,2006.

[14] Yanping Chen, Robert L. Probert, D. Paul Sims. Specification-based Regression Test Selection with Risk Analysis[C]. CASCON '02 Proceedings of the 2002 conference of the Centre for Advanced Studies on Collaborative research:Page 1.

[15] An Yong-Xin, Chen Yong-Zhang. Research on Risk-based Testing of Web Application. Chongqing: Chongqing University,2002.

[16] Chen Song-Yu, Chen Zhen-Ning. An Empirical Study on Cluster Test Selection Approaches. Nanjing: Nanjing University,2013.

[17] Rothermel G, Untch R H, Chu C, et al. Prioritizing test cases for regression testing[J]. Software Engineering, IEEE Transactions on, 2001, 27(10): 929-948.

[18] Chen Xiang. Survey of Test Case Prioritization techniques for Regression Testing. Journal of Software,2013.