Research on automatic generation method of trusted test scenarios based on behavior declaration

Xuejun Yu, Liping Wang
Beijing University of Technology, BeiJing, China

Abstract. In today's Internet era, the development of application software gradually becomes mature, which makes people's working mode also change accordingly. Applications, however, the scale is expanding constantly, make the internal structure of software is more complex than before, the application of the environment has become more open, causing the site to be attacked events occurred frequently, makes the credibility problem of software, especially in security, reliability, availability, and so on several aspects. At present, the research on software credibility is mainly based on application demand and software behavior, which are closely related to software credibility. Under this architecture. However, too many researchers focus on whether the application system meets the application requirements, and do not focus on the credibility before the system design. Therefore, this paper mainly takes the behavior statement as the entry point to study whether there are some untrustworthy factors before the system design.

Keywords: Behavior statement, Trusted test, Trusted test case, Automatic generation of activity graph.

1. Introduction
In this paper, the behavior declaration is used as the entry point, the activity graph is used to map the behavior declaration, and a test case generation method is proposed to generate more and less redundant test cases. Then, the key problems in the test method were studied, and the method and process of behavior declaration analysis were proposed. According to the results of behavior declaration analysis and the relevant knowledge of ant colony algorithm, the test cases were generated automatically, so as to generate the corresponding test cases. Finally, the software credibility evaluation system is designed and implemented. In terms of function realization, the system supports the definition and generation of behavior statement. On this basis, test cases can be generated automatically.

2. Research framework

2.1. Trusted Behavior Statement
The core of the whole life cycle credibility assurance model is the idea of "words consistent with deeds". In this model, the part of "words" refers to the static content in the whole life cycle process, that is, demand analysis, system design and implementation, software credibility design in the system judgment
environment. The "line" part refers to the dynamic content in the whole life cycle process, i.e., installation, on-line operation and uninstallation; "Consistency" is the process of verifying whether the "words" and "lines" of the system under the model are consistent[9].

In the framework of credibility testing method, the part of "speech" in the thought of "word consistency" should address the content, time point and way of "speech" definition. In this paper, the content of "speech" is defined from the two aspects of the trusted attributes extracted from the product and the activity diagram. From the perspective of the product, the behavior declaration of software behavior should be defined from the two perspectives of functional requirements and non-functional requirements. For example, the monitoring of user permissions requires that users attempt to access more times than their authorized scope, modify file permissions, and declare. In terms of trusted attributes, the activity graph needs to be transformed into a directed graph to analyze which path positions are sensitive areas and which sensitive areas may be damaged by bad user behavior. According to the analyzed sensitive areas, the trusted attributes should be mined.

2.2. UML modeling for requirements analysis

In software engineering, requirements analysis is carried out before system development. Requirements analysis refers to the work to be done to describe the definition, scope, purpose, and function of a new system when creating or modifying an old system. Demand analysis is a very important link in software engineering. In this process, the first thing to determine is the demand of customers. Only after determining the demand, can we analyze and find out the method to establish or modify the new system. The task of the requirements analysis phase is to identify all the functions of the entire software system[5].

UML is a modeling language, its definition includes two parts: semantics and notation. UML semantics refers to metamodel based definitions. The UML notation defines notation and provides a standard for developers to model systems using graphical notation and textual syntax[8].

2.2.1. Class Diagram. UML class diagram, showing the static structure of the system, its establishment process is as follows: First, determine the process to be modeled and the various classes involved in the process; Then the flexion and operation in each class are determined according to the specific working state of the system. Finally, a complete class diagram is constructed according to the association relationship of each class.

2.2.2. Activity Graph. UML activity diagram is one of the diagrams that model the dynamic aspects of an application system. It describes the dynamic behavior of an object over time by building a life cycle model of a class object. It is used to describe the behavior of instances of model elements, to describe the order of states and actions, to show the states that an object has, and to show how events affect those states over time. The establishment process is as follows: firstly, the state of each class in the class diagram should be determined according to the specific working process of the system; Then according to the specification of the specific description of the working process, determine the conditions of the mutual transformation of each state of the class

2.2.3. Sequence Diagram. Sequence diagrams, used to show relationships between objects, emphasize the chronological order of messages between objects, and show interactions between objects. The establishment process is as follows: firstly, the object of interaction and the scene described are determined; Secondly, find out the corresponding classes and their operations and attributes from the extracted type template. Third, create a sequence diagram for the classes and their operations and properties listed in the previous step.
2.3. Semantic model design

2.3.1. Activity diagrams are translated into meta-semantic analysis of code. This paper mainly studies the automatic generation code driven by UML activity diagram, and compares the activity diagram with the behavior statement according to the depth-first cycle. Therefore, before studying the automatic generation strategy of UML code, it is necessary to carry out semantic analysis on the primitives of UML activity diagram to distinguish which can automatically generate code and which cannot. If you can generate code, what is the most reasonable format to generate code in? The primitives of an activity graph are divided into three categories: nodes, edges, and other graphic elements of an activity graph.[1]

The tree node relationship of the activity diagram is shown as follows fig 1:

![Fig.1 Tree node relationship of the activity diagram](image1.png)

A simple example of selecting a node is as follows fig 2:

![Fig.2 Is the translation method for selecting nodes](image2.png)

Select the node to translate to the subordinate form code
If(x0 == x1)
{
    Activity2();
}
If(x0 == x2)
{
    Activity2();
}

A simple example of a fusion node is as follows fig 3:
The fusion node can be simply translated into the following code:

```java
If (x1 || x2) {
    Activity ()
}
```

The initial node is an active start node with no specific translation content.

Both the activity termination node and the flow termination node represent the end of the process. The difference is that the activity termination node terminates all behavior within the activity, while the flow termination node terminates a single class.

The edges of activity graphs can be divided into two categories: control flow, which represents the sequence of activities and does not translate; Object stream, translated as parameter passing, a simple example of the parameter is as follows fig 4:

Fig.4 Code table corresponding to parameters of activity diagram

2.4. Automatic generation of trusted test scenarios

Depth-first walk, just as its name implies, this kind of traverse method is based on the depth priority search or traversal of figure, as to what is the depth of preferential conditions, look at the following basic steps of DFS: DFS: starting from the current node, to mark the current node, then look for adjacent to the current node, and not tagged nodes:

(1) If there is no next node on the current node, return to the previous node for DFS
(2) If there is a next node on the current node, DFS will be performed from the next node
According to the above content, the activity graph will be transformed into a directed graph, and the safety factor is set for each node. The graph will be traversed from the beginning node, traversed according to the depth, divided into different scenarios when branches are

3. Trusted behavior declaration

3.1. Activity diagram coverage rules

UML activity diagram is to use case, describes the realization of function or functions depend on the order relation, namely the execution flow of a use case, this paper plans to use activity diagrams to cover the thought of rule, activity diagrams is to use case, describes the realization of function or functions depend on the order relation, namely the execution flow of the use case.It is mainly divided into the rules according to the activity graph itself and the coverage rules when it is transformed into a directed graph[4].

The rules for the activity diagram itself:

Each activity has only one starting node and at least one ending node. For the initial node and the end node, the constraints must be defined to represent the initial or end state of the activity.

In addition to the initial node and the termination node, all other nodes have at least one input edge and one output edge, thus ensuring that every node in the graph is reachable, that is, there are no isolated nodes.

Activities must indicate their input and output parameters, which represent the input and output data of the system or the state conditions of the system.

For control nodes, transition conditions on the input or output edges need to be identified as constraints.

Directed graph coverage rule:

Node override means that all nodes on the activity graph execute at least once.

Transition coverage means that all transformations on the activity diagram must be performed at least once.

Basic path coverage means that all paths on the activity graph must be executed at least once. For a loop, the path generated is called a basic path by not entering the loop and entering the loop once.

According to the credible basis of UML activity diagram in the early stage of software life cycle, the software credibility analysis method oriented to UML is used to analyze and compare, and the corresponding violation number is obtained. The corresponding trust level is determined according to the trust requirements of the software. The following table 1 defines an example of the trust level according to the number of violations

| Confidence level | Infractions |
|------------------|-------------|
| 0.25             | Infractions >=100 |
| 0.5              | 50<= Infractions <=100 |
| 0.75             | 10<= Infractions<=50 |
| 1                | Infractions <=10 |

After obtaining the trust index, some mathematical operations can be carried out to obtain the value of the credibility of the software. This series of mathematical or logical operations is called the software trust measure. Assume that each metric has an ID number and the metric value is trusti, where I is the metric's ID number. Assuming that the program has a total of n credibility measures, the credibility of the program can be composed of n sub-measure index values. This paper argues that the severity level of each measure is not the same, so the measurement index value of each different risk level should show different weight values. All metrics can be in the system according to the different risk categories and are rated level, this paper is divided into four different levels it (for different systems can be reset
to its), from level 1 to level 4, severity decreasing step by step, from important level decreasing weight coefficient respectively a1, a2, a3, a4, can attain: a1, a2 and a3 + a4 = 1, so each metric measurements: t(I) = x trusti ai, the credibility of the entire program t as follows:T= ∑T(I) Ni=1, where I is the sequence number of trusted indicators, and N is the number of trusted indicators.

3.2. Generate trusted behavior declaration mode

In the process of generating the trusted behavior declaration template, according to the specific requirement variables, the trusted attribute is extracted as the activity node, and the activity graph coverage rule is used for processing.

(1) The action node in the activity is transformed into the node of the test scene. Contains the activity ID, the activity name, the activity itself meets the conditions (such as the first node has no input edge, etc.), the activity node, the weight, and is transformed into the ID, name, condition, description, and safety factor in the trusted test scenario.

(2) Conditions under which edges in the activity diagram are transformed into test scenarios.

The activity diagram contains the edge ID, the edge name, the edge source node, and the edge target node, which are translated into the ID, name, source node, and target node in the trusted test scenario.

4. Generation of trusted test scenarios

This paper plans to transform the UML activity diagram into a directed connected graph based on the activity state, data on the edge and conditional relations, and then use the depth-first algorithm to traverse from the root node of the directed graph to generate trusted test cases. During traversal, the coverage criteria and validity of the generated trusted test scenarios should be considered.

4.1. The XML file describes the activity diagram

Firstly, an XML file is defined to describe an activity diagram, and then an activity diagram is transformed into a tree using DOM technology. Here is a simple example of an activity diagram XML file format, as shown in Figure 5:
4.2. Data structure
Node;
{
    int id;
    String kind;
}
Edge data structure:
struct edge
{
    int source;
    int target;
}
Among them, the id item in node represents the primitive sequence number of node in the activity diagram, and the kind item represents the type of node. The source and target items in edge edge represent the serial number of source node and destination node of edge respectively.

4.3. Parsing XML files
XPath is a language for finding specific information in XML documents and can be used to traverse elements and attributes in XML documents. If you need to extract information from an XML document, the simplest way is to insert an XPath expression into your program. Parsing XML with XPath takes the following steps: create a parse factory, create a parser, read a file through a parser, generate the W3C.dom.Document image tree, create an XPath object. The following code:

```java
DocumentBuilderFactory documentBuilderFactory=DocumentBuilderFactory.newInstance();
DocumentBuilder builder=documentBuilderFactory.newDocumentBuilder();
Document document=builder.parse("conf/55.xml");
XPath=XPathFactory.newInstance().newXPath();
String idPath="/Model/@id";
String id=(String) XPath.evaluate(idPath, document, XPathConstants.STRING);
System.out.println("id="+id);
```
Through XPath, the XML file is parsed to obtain node attributes and node names.

4.4. Contrast the imported XML with the behavior declaration
According to the behavior declaration template information generated above and the anti-depth-first traversal algorithm, a credible test scenario is formed and a report is automatically generated. The specific contents are as follows:
The corresponding nodes are compared with the nodes on the behavior declaration template according to some activity diagram coverage rules, as shown in Figure 6.
Fig. 6 Activity diagrams cover rules compared to action statements

Build an automatic report generation environment, and automatically generate reports with the behavior life template. Case directory decentralizes test scenarios: each test scenario asserts the final result and the expected result; Delegate reporting; Run_all_case write to execute all path scripts, file directory as shown in Figure 7, run run_all_case.py, and generate report report under report directory as shown in Figure 8 [7].

Fig. 7 file directory
5. Conclusion

This paper has completed the generation of trusted automated report driven by activity graph, but there are also many shortcomings. For example, there is not much research on the selection of algorithm for path traversal, only one method is selected for traversal, and further research in this area is needed in the future.

References

[1] Chen Ci. Research on software Requirement Extraction Technology based on Semantic Analysis [J]. Ship Electronic Engineering 2020,6:118-123

[2] Liu Yuebo. Semantic analysis of UML sequence diagrams based on referential semantics [J]. Journal of chifeng university 2012,28(12):33-35

[3] Li Xin. XML Keyword query method based on semantic tree Model[J].Yang zhou university,2013,16(3):52-56

[4] Feng Tingting. Research on test Case Generation based on UML Activity Diagram Model [D].Tianjin: Nankai University,2020

[5] Jiang Dongyu, Kang Dazhou, Wang Shun. Research on Ontology-based Demand Analysis and Software Architecture Design [J].Computing technology and automation, 2017,036 (02) : 129-135.

[6] Li Daiyi. Research on knowledge Construction and Recommendation of Case Software Non-functional Requirements [D].Kunming: Yunnan University, 2019.

[7] Liu Miao. Research and application of Rest-style Software Credibility Test based on Behavior Declaration [D].Beijing: Beijing University of Technology, 2017.

[8] Liu Xinyu. Formal Analysis method of System Requirements based on UML [D]. Beijing: Beijing Jiaotong University,2015.

[9] Yu Xuejun, Xiao Ran. Research on Credibility Test Method and Credibility Calculation Based on Behavior Claims. Beijing University of Technology, 2018.