Research on Generating Method of Embedded Software Test Document Based on Dynamic Model

MingCheng Qu¹,²*, XiangHu Wu¹,², YongChao Tao², Ying Liu¹,²
¹School of Computer Science and Technology, Harbin Institute of Technology; Harbin 150001 Heilongjiang China
²Shenzhen Academy of Aerospace Technology, Shenzhen 518057 Guangdong, China

*Corresponding author e-mail: qumingcheng@hit.edu.cn

Abstract. This paper provides a dynamic model-based test document generation method for embedded software that provides automatic generation of two documents: test requirements specification documentation and configuration item test documentation. This method enables dynamic test requirements to be implemented in dynamic models, enabling dynamic test demand tracking to be easily generated; able to automatically generate standardized, standardized test requirements and test documentation, improved document-related content inconsistency and lack of integrity and other issues, improve the efficiency.

1. Introduction
At present, China's embedded software research and development, testing and process management is still the traditional document-driven software development methods, the use of a large number of human needs and design documents, and hand-written code and test cases, while ensuring the quality of software to use a large number of Manpower to review the software and code generated at all stages of the software development, walking and testing. The preparation of requirements specifications documents, design documents, test requirements and other documents have become the most troublesome thing programmers. The following questions are currently written:
(1) The document is not complete; the relevant content is inconsistent, lack of integrity;
(2) Human omission, bringing human error;
(3) The information stored in the document on the "man-made" test activities cannot be continuous, mandatory, complete, regular constraints and guidance;
(4) Document storage information does not maximize the use of the previous stage of the output results and so on.

In this paper our objective is to provide a dynamic model based test document generation method, automatically generate standardized, standardized documents and improve work efficiency.

2. Literature Survey
The quality assurance effect of the software depends on the accuracy of the document's description of the requirements and design and the degree of detail of the reviewers [1, 2]. However, the inaccuracies of natural language and the shortcomings of manual management are difficult to avoid omission[3, 4], which makes the software development method based on a large number of human work, non-automatic, document-centered software development method is low efficiency, high cost, , Quality is
difficult to guarantee and other issues, seriously hampered China's aviation, aerospace, automotive and other industries.

3. A Method of Generating Embedded Software Test Document Based on Dynamic Model

As shown in Figure 1, the steps of the test document generation method based on the dynamic model in this paper.

3.1. Develop document templates with Freemarker syntax tags

Save the Word document template as an XML file and make the Freemarker tag, resulting in a document template with the format .ftl. Freemarker tags are as follows:

(1) loop markup: `<# list varList as vartemp> content </# list>`

(2) Normal tag: `$ {propertyName}`

Where varList represents the data set to be looped, vartemp represents the current data in the loop, propertyName represents the variable, and the normal tag can be used in the loop tag. For example, if vartemp is an object and name is its attribute, you can use `$ {vartemp.name}` to display the value attribute of vartemp.

The format requirements for editing Word document templates are:

(1) All items in the form must belong to the same cycle level;
(2) Pictures to have a common mark, the contents of the ordinary mark for the image of the path;
(3) Each common tag and its circulation level has a one-to-one relationship, can not read the elements of the content across the loop level;
(4) The end of the cycle mark must be WORD carriage return line, the general mark is not limited;
(5) The contents of the loop mark and the normal mark must be able to find the only correspondence in the data.

The format requirements for editing Word document templates are:

(1) All items in the form must belong to the same cycle level;
(2) Pictures to have a common mark, the contents of the ordinary mark for the image of the path;
(3) Each common tag and its circulation level has a one-to-one relationship, can not read the elements of the content across the loop level;
(4) The end of the cycle mark must be WORD carriage return line, the general mark is not limited;
(5) The contents of the loop mark and the normal mark must be able to find the only correspondence in the data.

Develop document templates with Freemarker syntax tags

Depth understanding of the software under test, using the state diagram, timing diagram, fault tree, causal and decision table and the nesting mechanism between the graphics and graphics extended semantics to build embedded software dynamic model to describe the embedded software dynamic test requirements.

The verification of the completed graphics model, according to the purpose of testing the test items and generate test items corresponding test cases, the use of the form of database storage test items and test cases.

Automatically generate test documents based on dynamic models

Fig.1 The steps of the test document generation method based on the dynamic model

3.2. Depth understanding of the software under test, using the state diagram, timing diagram, fault tree, causal and decision table and the nesting mechanism between the graphics and graphics extended semantics to build embedded software dynamic model to describe the embedded software dynamic test requirements. In addition, the graphics model is parsed and saved with an XML file tree structure.

The nesting mechanism between the graphs is as follows:

(1) Data flow graph data processing can be nested timing diagram;
(2) The state nodes of the state diagram can nest state diagrams, data flow diagrams, fault trees, causal maps, and decision tables;
(3) The bottom event of the fault tree can nest the child fault tree.
The extended semantics of the graph are as follows:

1. The state diagram of the entity to add the necessary test semantics, such as the state node to add data variables, nested graphics nodes and other attributes;
2. The elements of the timing diagram add the necessary test semantics, such as simple messages to add monitoring conditions, message content, message type, message delay and other attributes;
3. The fault tree's primitive adds the necessary test semantics, such as the bottom event that adds the CMP attribute (quantitatively describes the occurrence of the bottom event);
4. To determine the table to add the necessary test semantics, such as according to the logical expression automatically generate a decision table, add the table to determine the pre-conditions.

3.3. The verification of the completed graphics model, according to the purpose of testing the test items and generate test items corresponding test cases, the use of the form of database storage test items and test cases.

According to the criteria of the verification of the graphics model, the completed graphics model is verified, and the criteria for the verification of the graphic model are as follows:

1. State diagram only one initial state, the state does not allow the existence of two or more direct transfer;
2. The object of the timing diagram from the data flow diagram of the data processing or other elements;
3. Event constraints of the fault tree (a fault tree has only one top event, the bottom event must be quantified, etc.);
4. Causal map of the constraints (a causal map only one main reason);
5. Determine the conditional element action element combination constraint of the table.

3.4. Automatically generate test documents based on dynamic models

The test document is a Word document and the test document is generated based on the Freemarker template engine. Apply the "Template + Data Model = Output" formula to get "Test Item + Test Requirement Specification Document Template = Test Requirement Specification Document", "Test Item / Test Case + Configuration Item Test Description Document Template = Configuration Item Test Description Document". Therefore, replace the syntax markup in the document template with the data (based on the graphics derived from the graphics model, the collection of test items, and the test case collection) to get the required test documentation. The specific algorithm description is shown in Algorithm 1 and Algorithm 2.

Algorithm name: Algorithm 1
Input: Dynamic model, test requirements specification document template
Output: Test requirements specification documentation
Process:
1) To obtain a dynamic model;
2) To obtain the name of the software under test and dynamic model derived pictures;
3) Read all types of test items from the database, including functional test items, interface test items, performance test items and margin test items, and obtain the priority description of the test items;
4) Resolve the dynamic model, from the first layer of data flow chart nested traversal, access to all dynamic map node name and description.
5) Based on FreeMarker load test requirements specification document template file;
6) set the output document path and name, and create a test requirements specification document file;
7) Replace the syntax tags in the document template with the data obtained from 2, 3, 4 to get the required test documents.

Algorithm name: algorithm 2
Input: Dynamic model, test description document template
Output: Configuration Item Test Description Document
Process:
1) To obtain a dynamic model;
2) To obtain the name of the software under test;
3) Read all types of test items from the database, including function test items, interface test items, performance test items and margin test items, and obtain the test items corresponding to the test items collection;
4) Based on FreeMarker loading configuration item test description document template;
5) Set the output document path and name, and create a configuration item test description document file;
6) Replace the syntax tag in the document template with the data obtained by the 2, 3 to get the required test document.

4. Experimental Results
Experiments show that the method of generating test documents based on dynamic model proposed in this paper is feasible. It can generate the specification document according to the template. This method can reduce the errors caused by manual writing, improve the inconsistency of document content, and improve work efficiency.

5. Conclusion
In this paper, an idea is implemented to propose a test document generation method based on dynamic model on the basis of dynamic models (state diagrams, timing diagrams, fault trees, causal maps, and decision tables). Compared with the prior art, the method proposed in this paper has the following advantages:

1) When creating a dynamic test model for embedded software, the test designer will find some inconsistencies or incomplete information in the original software requirements or system model, so that more errors can be found early in the embedded software lifecycle.
2) Can easily achieve dynamic test requirements tracking, model reusability reduce duplication of effort, improve efficiency;
3) the use of Freemarker grammar mark with the document template, to achieve frequent changes in content and format data preservation, which greatly facilitates the user at any time to modify, without worrying about the operation of the program;
4) To improve the document content is inconsistent, because the model changes, you can re-generate the test document;
5) Automatically generate standardized, standardized documents, reducing the errors caused by manual writing, improve work efficiency.

Acknowledgments
This work is funded by National natural Science Foundation of China (No.61402131); China Postdoctoral Science Foundation Special Fund (2016T90293); China Postdoctoral Science Foundation Item (No.2014M551245); Heilongjiang Provincial Postdoctoral Science Foundation Item (No.LBH-Z13105); Central University Basic Scientific Research Business Expence Special Fund (No.HIT.NSRIF.201651); Harbin Science and Technology Innovator Special Fund (2015RAQXJ047). Science Innovation Committee technology development project (NO.CXZZ20150504150427057)

References
[1] Manisha Sahni, Isha Sharma, Jasleen Kaur. Analysis of Test Case Prioritization Techniques[J]. International Journal of Computer Science and Mobile Computing, 2014(3): 917-921.
[2] Parampreet Kaur, Gaurav Gupta. Coverage Analysis and Chinese Postman Algorithm for Efficient Model-Based Test Generation[J]. International Journal of Computer Science and Mobile Computing, 2013(2): 491-494.
[3] Z. Havlice, V. Szabóová, J. Vízi. Critical Knowledge Representation for Model-Based Testing of Embedded Systems[C]/SAMI 2013 • IEEE 11th International Symposium on Applied Machine Intelligence and Informatics. New York: IEEE, 2013: 169-174.
[4] Gerson Sunyé, Eduardo Cunha de Almeida, Yves Le Traon, Benoit Baudry, Jean-Marc Jézéquel. Model-based testing of global properties on large-scale distributed systems[J]. Information and Software Technology, 2014(56): 749-762.