Design of the Visual Model Platform of Testing System Based on GMF

X.M. Liu
College of Information, Beijing City University
China

Y.P. Liu
College of Computer, Beihang University
China

S.M. Liu
College of Science and Information
Qingdao Agriculture University
China

J. Wu
College of Computer
Beihang University, China

***Abstract*—Modelling the testing system brings the idea of model driven into testing field. Reusing testing resources is the core of the model, which effectively shortens the testing cycle. When the testing requirements or design specifications of the system under test (SUT) are changed, the testing system designers can adapt the change by adjusting the testing model. Based on the definition of the meta-model in this paper, the testing system model and instance data editor have been established by GMF (Graphical Modelling Framework) based on the mode driven development method. It realizes the visual modelling and the automatic generation of a part of code. In this paper, the situation of GMF is firstly introduced. Then it describes the process of designing and developing the visual modelling and the method of researching the visual modelling and automatically generating a part of code of the testing system in detail. At last, it presents the process and results of visual modelling by a website.**

Keywords-model-driven testing; graphical modelling framework (GMF); visual modelling; testing system

I. INTRODUCTION

With the change and development of application systems, in order to meet the demands for software engineers to ensure the high quality of the application system, the software testing technology is also thriving. It has gradually developed from the initial manual testing to semi-automated testing or automated testing by using various testing tools. In recent years, the MDT(Model-Driven Testing) is rising, with the improvement of the description function of the visual modelling tools(such as UML, Unified Model Language) and the advent of various functional testing tools, the application of the methodology of MDT in modern software system has been accelerated [1]. This paper mainly implements the visual modelling of testing system based on GMF, which makes the testers abstract the test points and test processes that are noticed and directly expressed them without ambiguity. At the same time, it provides the methods to generate TTCN-3 code automatically which can be used to transform the testing system model into TTCN-3 code automatically [2].

The earlier graphical interface application programs were developed by use of GEF (Graphical Editing Framework), such as the JBPM to define the process, which adopts the combination of GEF and JavaBean, its disadvantage is that the process on model layer is too cumbersome, which needs developers manually implement the model notification mechanism, and is not conductive to the later expansion and maintenance. Once the requirements change, the model structure is difficult to be mended [3]. EMF (Eclipse Modelling Framework) helps us develop Eclipse applications by model driven approach. The combination of EMF and GEF has gradually become the main mode for constructing GEF applications. The GMF framework can solve the EMF and GEF integration difficulties, so that we can use more simple methods to achieve the applications developed by EMF and GEF at the same time before. And various problems brought by the combination of the two need not be thought. At the same time, with the help of Eclipse plug-in mechanism, GMF also provides rich scalability to develop applications that are suitable for specific needs. The two command mechanisms are connected with each other by the adaptive way. It also makes the expansion of many functions when integrating the two frameworks [4].

II. VISUAL MODELLING BASED ON GMF

A. The Design Mode

Designing visual modelling environment mainly uses the layered design pattern of MVC (Model-View-Controller). At the same time, in order to support the model storage function, the model storage /loading layer is increased under the model layer. Therefore, the model editor is divided into four layers, followed by the storage /loading layer, the model layer, the controller layer and the view layer, shown in figure. Ⅰ

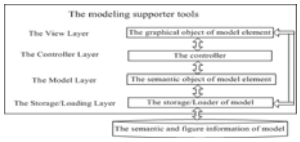


FIGURE Ⅰ. HIERARCHICAL CHART OF VISUAL MODELLING ENVIRONMENT.

Layer of Storage/Loading. The model storage supports the function to store models and the semantic and graphical information of models. The model loader can load the information into the system, so that the users can alter and edit them. Layer of Model. The semantic object of model element

© 2015. The authors - Published by Atlantis Press

lies at the Model Layer. It only maintains the information related with the modeling element semantics. It doesn't care its graphical show, such as its shape, color, size and layout. Layer of View. The graphic object of model element lies at the View Layer [5]. It only maintains the information related with the graphical show of modeling element, such as the above-mentioned shape, color, size and coordinates, etc. Layer of Controller. In the MVC model, the semantic object of model element and its graphical show can't directly control each other. The operation is completed by the controller located between them. The main role of the controller includes the following aspects: (1) it receives the information from the View Layer and processes the Model Layer. For example, if a user wanted to add a sub-model element into a model element, the mouse click event from the user command in graphical element of the parent model will be parsed as the corresponding request to create a sub-model element by the controller. According to the request, the sub-model can be created in the semantic object of its parent model. (2) It reflects the change of Model Layer to the View Layer. For example, after the semantic object of a sub-model element is added into the semantic object of its parent model element, the corresponding graphic object will be created by the controller and displayed in the view [6].

B. Developing GMF Model Editor

GMF is a plug-in of Eclipse. It supports to develop a visual model editor. GMF consists of two main components. They are tooling components and runtime components.

1) Tooling components: It consists of a series of editors which support to create the Domain Model, Graphical Definition, Tooling Definition and the Mapping Model that combines the front three models. The Domain Model, Graphical Definition and Tooling Definition are the generalities of visualization model editors. The Domain Model represents the semantics of the model. It equals to the Model Layer of fig.1. The Graphical Definition is a graphical show. It equals to the View Layer of fig.1. The Tooling Definition defines the tool to create the model element for each model element. The Mapping Model combines the model element of the Domain Model with the Tooling Definition and the Graphical Definition. It is equal to the Controller Layer of fig.1. The Tooling components can be used to further generate the visual model editor [7]. The process is shown as figure. Ⅱ.

2) Runtime components: The model editor plug-in created by the Tooling Components relies on the Runtime Components. It provides the basic components shared by all the model editors. The basic components and the components related with specific model in the model editor plug-in constitute an executable model editor in common. On the basis of the model editor plug-in generated by GMF, the developers can further customize and modify the model editor according to their own needs.

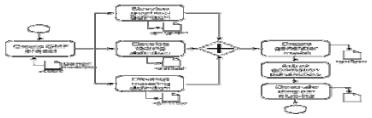


FIGURE Ⅱ. DIAGRAM OF THE PROCESS TO GENERATE GRAPHICAL EDITOR ON GMF.

III. IMPLEMENTING THE VISUAL MODELLING

A. The Architecture of Visual Modelling Platform

The creation of the visual modelling tool and TTCN-3 code is based on the definition of testing meta-model, shown as figureⅢ.

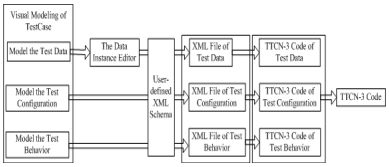
The test data model, the test configuration model and the model of test behaviour can be created by the visual modelling tool based on GMF. The data instance editor can be generated by the test data model. It can edit the specific test data as the data pool required by the test. In order to generate TTCN-3 code, the XML schema corresponding to TTCN-3 code is firstly defined. The test data model and the data content of the data instance editor can be used to generate the XML file of test data. The test configuration modelling can be used to generate the XML file of the test configuration.

The test behaviour model can be used to generate the XML file of the test behaviour. Then the XML file of test data generated the test data TTCN-3 code, the XML file of test configuration generates the test configuration TTCN-3 code, the XML file of test behaviour generates the test behaviour TTCN-3 code. Finally the TTCN-3 code of data, configuration and behaviour are integrated to generate the complete TTCN-3 code.

B. The Meta-Model of Testing System

The definition of the meta-model of testing system refers to U2TP with the .ecore file of EMP. It automatically generates the corresponding code using EMP as the basis of building the visual modelling tool. The meta-model specifies the elements of model and the semantics among them. It includes the test data meta-model, the test configuration meta model, the test behaviour meta model, which describe the test system from data, static structure and dynamic behaviour respectively.

The test data meta-model defines a set of concepts describing the test data. It mainly includes DataPool, DataParticition, DataInstance and various complicated data structure. The test configuration meta model describes the static structure of the testing system. It includes TestCase, TestSenario, Timer, SUT, TestComponent, TestInterface and TestInterfaceConnection. The test behaviour meta-model describes the behaviours of the testing system. It includes StartTimeAction, StopTimeAction, StimuliAction, ResponseEvaluateAciton, the types of branch action and the Verdict of TestingResults. The Testing results define four kinds, they are pass, fail, error and inconclusive.



FIGUREⅢ. MODEL THE TESTING SYSTEM AND CREATE TTCN-3 CODE.

C. *The Testing System Model*

It is a technical difficulty how to build the data model, configuration model and the behaviour model based on TTCN-3 and U2TP.

Data Modelling is reflected into two respects, namely the data type and the specific data. The data type includes SimpleType (such integer, float), Enum, Record, Recordof and Setof. In the specific data, DataInstance is an instance object of data type. It includes the specific test data value. DataInstance editor can edit and store the test data. The attribute describes the dependency between data types. The role of configuration modelling is to define the test suite and the connection with the SUT. Firstly, the test suite is defined. Next, the interface between them will be defined. The connection with the SUT interface will be defined at last. The type of the interface is also be defined, which represents the message type.

During the behaviour modelling, the test suite is firstly defined. Then the interface between them will be defined. The test behaviour includes StimuliAciton, ResponseEvaluation, StartTimeAction, StopTimeAction, AltAction, WhileAction, ForAction, IfAction. The test behaviour model is shown as UML sequence chart. The Action is the test suite and the SUT. The message is the function call and the send-receive message between the test suite and the SUT. The position from top to bottom indicates the sequence of test actions. The composite fragment represents the complex control flow which can show the branch, the circulation and the reference etc. The TimerAction is represented by the graphical element from TTCN-3 figures.

D. *Creating Test Script Automatically*

The test model can be converted into the XML file according to the corresponding schema. Then the XML file should be parsed. At last it is transformed into TTCN-3 code. At the same time, the corresponding data instance editor is created by use of data model. By editing the data in the data instance editor, the specific test data values can be provided for the TTCN-3 data module.

During the transformation from the test model to TTCN-3 code, the mapping between U2TP meta-model and TTCN-3 meta model must be defined firstly. Then the elements of the test model are read and transformed into the corresponding XML elements. If it is a sub-element of an element, it is also expressed a sub-element in XML, the attributes of the model element are expressed as the XML attributes. For the test model, the sequence between the test behaviours is essential. Therefore, the sequence maintains the consistency in XML. The conversion from XML to TTCN-3 code needs refer to TTCN-3 core language.

IV. CASE STUDIES

The following example to test a campus website login function verifies the method and the process of visual modelling based on GMF. The testing system sends username and password to the campus website. Then the test determination will be performed according to the feedback data from the website.

A. *Visual Modelling*

The test data model describes the test data type of the testing system. It includes a record type named LoginRequest and Enum type named LoginResponse, they respectively indicates the login request and the login response. The LoginRequest includes two string types named username and password. The LoginResponse includes three Enum types named SUCCESS, WRONG-PASSWORD and UNEXPECTED.

The test configuration model includes three components named Client, NS and rootDNS. Every component has an interface connecting with the SUT interface, whose type is DNSPort.

The test behaviour model includes three test components and the SUT. The main test component named Client sends a message queryA to the SUT, then starts running and starts a reference to another test scenario. According to the message received from the test component, the branch decision is formed. The No.1 branch, if receiving response, stop running and the result is passing. The No.2 branch, if receiving others, stop running and the result is fail. The No.3 branch, if no receiving the message and the time is out, the result is inconclusive.

B. *Editing the Data Instance and Creating Ttcn-3 Code*

The data instance editor is shown as figure Ⅳ It can add, delete and update the needed data instance. It is stored as XML file. Before creating TTCN-3 code, the XML schema corresponding to TTCN-3 code has been defined, in order to be inconsistent with XML file which is used as the media.

V. SUMMARY

Model driven test is a modern test method making up the disadvantage of automatic test. This paper aims at the application of GMF visual modelling technology to model driven testing field. By using the Eclipse plug-in, the graphical application can be developed. It realizes the visual modelling and the automatic generation of TTCN-3 test script of the testing system.

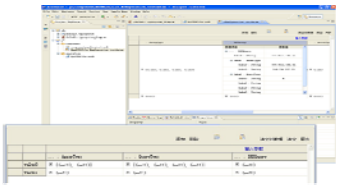


FIGURE Ⅳ.DATA INSTANCE EDITOR.

REFERENCES

[1] European Telecommunications standards Institute (ETSI), Sophia-Antipolis(France). ETSI European Standard (ES) 201 873 (2002/2003): The Testing and Test Control Notation Version 3(TTCN-3), Part5: The TTCN-3 Runtime Interfaces (TRI), 2003.

[2] I. Schieferdecker, G. Din: A meta-model for TTCN-3. 1st International Workshop on Integration of Testing Methodologies, ITM 2004, Toledo, Spain, Oct. 2004.

[3] OMG: UML 2.0 Superstructure Final Adopted Specification, www.omg.org/cgibin/doc?ptc/2003-08-02.

[4] J. Zander, Z.R. Dai. From U2TP Models to Executable Tests with TTCN-3 - An Approach to Model Driven Testing. Testing of Communicating Systems，2005.

[5] European Telecommunications Standards Institute (ETSI), Sophia-Antipolis (France). ETSI European Standard (ES) 201 873 (2002/2003): The Testing and Test Control Notation Version 3 (TTCN-3), Part 3 TTCN-3 Graphical presentation Format (GFT), 2003.

[6] European Telecommunications standards Institute (ETSI), Sophia-Antipolis(France). ETSI European Standard (ES) 201 873 (2002/2003): The Testing and Test Control Notation Version 3(TTCN-3), Part 6: The TTCN-3 Control Interfaces (TCI), 2003.

[7] OMG: UML 2.0 Testing Profile. Final Adopted Specification, ptc/04-04-02, 2004.