Test data generation for web application using a UML class diagram with OCL constraints

Shoichiro Fujiwara · Kazuki Munakata · Yoshiharu Maeda · Asako Katayama · Tadahiro Uehara

Abstract In this paper, we report on our current work toward efficient and effective verification of web application’s basic design. We use a UML class diagram with Object Constraint Language (OCL) to describe the application behaviors and data constraints. Then we generate test data from the formally represented specifications. We make the observation that key web application behaviors can be captured through table size constraints as well as data constraints like foreign key constraints. Based on the observation, we translate the OCL specification into the equivalent constraints using table size expressions. We present a scheme to generate test data from the translated constraints using a Satisfiability Modulo Theories solver. We employ two techniques to reduce constraints. The first is string handling and the other is decomposition of table structures. We also report on an experimental result of test data generation. The result indicates a potential that our scheme works well for real applications in reasonable times.

Keywords Web application · Basic design · OCL · Testing · Data generation

1 Introduction

Current trends indicate that more and more projects are employing web technologies in the development of business applications. Further, projects often face a shortage of resources such as budget, time, engineers, etc. This often results in having to release a system without enough testing or verification, and the system may constantly cause program failures in actual operation. This is one of the main reasons why there exists a higher expectation for an efficient and effective verification technology of web applications.

In this paper, we describe our practice to meet with the expectation. We focus on the basic design of web applications, which specifies system functions to implement. The basic design is worth verification because it bridges programs and system requirements. Another reason to choose basic design is its closeness to the implementation. It enables us to construct a concrete model as the basis for verification activities. In this work, the concrete model, or specification, is formally written with UML and Object Constraint Language (OCL). The UML specifies system entities, while the OCL describes system behaviors and relations between the entities. We try to accelerate testing, a major verification activity, with the UML and OCL specification. From the view of testing, specifications can be regarded as constraints which test data and expected values should comply with. This paper is about test data generation from the specification.

The rest of this paper is organized as follows. Section 2 describes what is specified in the basic design of web applications. In Sect. 3, we show our web application modeling with UML and OCL. Section 4 introduces the basic concept of test data generation from the model. Section 5 is the detail of the test data generation scheme followed by an experiment. We discuss practical issues in Sect. 6. Section 7 refers to related work. We conclude this paper in Sect. 8.
2 Basic design of web applications

Things specified in the basic design of a web application include logical database models, data items on each screen, events and behaviors around each event. An event stands for a unit of application processes. One example is a screen transition or a business logic execution, which is usually triggered by a user action like clicking a button. A behavior of the web application refers to a program state transition caused by an event. A behavior consists of two program states of a pre-event state (pre-state) and a post-event state (post-state). Each program state is composed of a database state and a screen state and can be represented by its satisfying conditions. In this paper, we focus on behaviors of a web application, provided that the structure of database tables and screen items are given. We also assume that it is known what events can be triggered on each screen.

Let us give an example web application as Fig. 1. This application is a shipping management system. A user searches products by a Product Code on the Product Search screen. The Search button triggers a search event in which a query is sent. The query is to select all records from the Product Master table where the Product Codes start with the Product Code on the previous screen. The records are sorted in the ascending order of their Product Codes. After the event, the screen transitions to the Search Result screen which displays the query result on the Product List from top to bottom. The user then selects a product to ship from the Product List and clicks the Ship button to issue a ship event. The next screen is Shipping, in which the user selects a Warehouse and inputs Shipping Qty (Quantity) of the product. Next, the user triggers a determine event with the Determine button. The application checks whether there are enough Qty In Stock of the product in the selected warehouse. The screen after the event is Transaction Result. If there is enough Qty In Stock in the warehouse, the Result Code would be 10001. If not enough, the code is 20001. When there is no entry of the product in the warehouse, 20002 is assigned to the code.

Now we go into the details of the behaviors around the determine event. There exist three behaviors, which are specified as below:

Behavior 1: If (pre-condition) “there exists a record in the Inventory Status table such that Whse Name and Product Code match the Warehouse and Product Code in the Shipping screen, and the Qty In Stock of the record is larger than or equal to the Shipping Qty in the screen”, then (post-condition) “the screen transitions to the Transaction Result and the Result Code on the screen is 10001”.

Behavior 2: If (pre-condition) “there exists a record in the Inventory Status table such that... (same as the behavior 1), and the Qty In Stock of the record is smaller than the Shipping Qty in the screen”, then (post-condition) “the screen transitions to the Transaction Result and the result code on the screen is 20001”.

Behavior 3: If (pre-condition) “there does not exist a record in the Inventory Status table such that... (same as the behavior 1), and the Qty In Stock of the record is smaller than the Shipping Qty in the screen”, then (post-condition) “the screen transitions to the Transaction Result and the result code on the screen is 20002”.

We also take a look into a behavior on the search event. Suppose the application cannot show the query result if the number of records in the result exceeds the display limit. We describe a normal case behavior now:

Behavior 4: In the search event, if (pre-condition) “the number of records in the Product Master table for which Product Code starts with the Product Code in the screen is not more than the display limit”, then (post-condition) “the screen transitions to the Search Result and the Product Code and Price of records in the Product List table from top to bottom equals the Product Code and Price of records in the Product Master table sorted in