Piacente Cristian 866020

Homework H9 – Model based testing

Software Quality, Academic Year 2023-2024, University of Milan – Bicocca

Consider the informal specification of a simple online book-store system.

- Identify independently testable features.
- For each of the identified features, identify a method to generate test cases and justify your choice.
- For each of the identified features, derive a reasonable model. Briefly comment the choices and complete the informal specification to produce the model.

On-Line bookstore

Users can access the platform to search, buy, and sell books.

When users buy some books, they can obtain a discount which depends on their status:
Users owning a “fidelity card” are entitled a 20% discount.
Users owning the “fidelity card” get an additional 5% discount if they have previously bought on the platform more than 100 books.
Users can trade a book they own when they buy a new one. In this case, they get a 20% discount.
If they also own a “fidelity card”, they get a 25% discount.
Students and professors get a 10% and 15% discount, respectively. This specific discount is applied only if the user is not eligible for other discounts (fidelity card, trading book).

When buying new books, users must provide a non-empty cart, their credentials, a shipping address and a valid credit card.

The system checks for the validity of such information. If the cart is empty, the process terminates. If the credentials are correct, the systems checks the validity of the registration: if the registration is still valid, the process continues. Otherwise, the system asks to renew the registration. If the shipping address is an international address, the system adds a shipping charge. Finally, the system processes the credit card data. If the provided expiration date is not valid, the system asks for new data. Otherwise, it sends the data to an external payment service and waits for an acknowledgement. If the external service communicates that the payment has been successfully completed, the process terminates. Otherwise, the system reports the anomaly and asks for a new credit card.

The books are stored on the platform according to the following DTD scheme:

```
<!ELEMENT books (book+)>
<!ELEMENT book (title, author+,year,rating+,review+,price-new,price-pre-owned)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT rating (0|1|2|3|4|5)>
<!ELEMENT review (#PCDATA)>
<!ELEMENT price-new (#PCDATA)>
<!ELEMENT price-pre-owned (#PCDATA)>
```

The solution should clearly indicate

- the feature, by giving a meaningful name referring to the specification
- the chosen model, for example finite state machine, decision table, grammar, ...
- the model obtained for the feature.

The specification lets us identify the following independently testable **features**:

- **Discount**
  The system applies discounts based on user status, such as fidelity card ownership, book trading, and status as a student or professor.
- **Payment**
  The system checks the cart’s contents, user authentication, payment validation, and transaction completion.
- **Book storage**
  The system stores the books with a defined DTD scheme.

Piacente Cristian 866020

Homework H9 – Model based testing

Software Quality, Academic Year 2023-2024, University of Milan – Bicocca

We can now **choose a model** for each feature.

- *Discount*
  We can notice that the discount part of the specification is expressed as a decision structure (conditions on input values and corresponding discount), so it's appropriate to choose a **decision table** model.
  We could apply the MC/DC adequacy criterion for efficiently generating the test cases, after transforming the conditions into boolean expressions.
- *Payment*
  We can extract a **finite state machine** model from the payment part of the specification, which is an informal natural language specification of intended behavior, to show the interactions between the system and its environment.
  We could use transition coverage criterion for the test cases generation, which means that each transition has to be traversed at least once.
- *Book storage*
  The last part of the specification describes a DTD scheme, which is a set of rules that constitute a **grammar**: it's reasonable to exploit the **BNF** notation.
  We could refer to a simple criterion to produce a set of test cases: each production rule has to be exercised at least once.

Let's **derive the models**.

Decision table for feature Discount

Fidelity	T	T	T	T	F	F	F	F
Above100	T	F	T	F	-	-	-	-
Trade	T	T	F	F	T	F	F	F
Stud	-	-	-	-	-	F	T	F
Prof	-	-	-	-	-	T	F	F
Out	30%	25%	25%	20%	20%	15%	10%	0%

Abbreviations

Fidelity	Owning a fidelity card
Above100	Having previously bought more than 100 books
Trade	Trading an owned book
Stud	Being a student
Prof	Being a professor

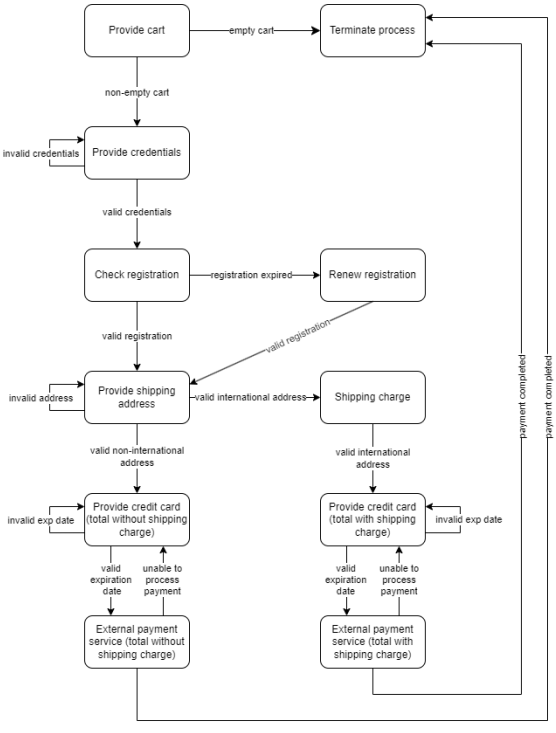
Finite state machine for feature Payment

To derive a finite state machine model, the specification has been completed with these additional details:

- If the credentials are not correct, the system asks for the credentials again.
- If the shipping address is not a valid address, the system asks for the shipping address again.

To cope with the "history sensitivity" issue, for the payment processing different states have been created in the model: those which process the payment without any shipping charge and those which consider a shipping charge. This makes the payment transitions depend on the path.

We could improve the specification more, for example by adding a maximum number of attempts to avoid infinite loops in the model (and for security reasons in a real-life system).



Piacente Cristian 866020

Homework H9 – Model based testing

Software Quality, Academic Year 2023-2024, University of Milan – Bicocca

BNF grammar for feature Book storage

Assuming TEXT is a terminal symbol which stands for text data (with whole numbers, whitespace, line breaks, and other reserved characters suitably escaped), here is a BNF grammar definition for the book storage feature:

<books> ::= <book> | <books> <book>
<book> ::= "<book>" <title> <authors> <year> <ratings> <reviews> <price-new> <price-pre-owned> "</book>"
<title> ::= "<title>" TEXT "</title>"
<authors> ::= <author> | <authors> <author>
<author> ::= "<author>" TEXT "</author>"
<year> ::= "<year>" TEXT "</year>"
<ratings> ::= <rating> | <ratings> <rating>
<rating> ::= "<rating>" ("0" | "1" | "2" | "3" | "4" | "5") "</rating>"
<reviews> ::= <review> | <reviews> <review>
<review> ::= "<review>" TEXT "</review>"
<price-new> ::= "<price-new>" TEXT "</price-new>"
<price-pre-owned> ::= "<price-pre-owned>" TEXT "</price-pre-owned>"