IN2013, Week 10 – Software Testing: Operational profile

Dr Peter Popov

Answers

29th November 2018

Scenario

Part 1: Operational profile in System Testing

Operational profile is an important concept in statistical software testing. Recall that the operational profile defines a ***probability distribution*** over the set of possible use cases. The probability P(A) associated with use case A will define the chances that a randomly chosen use case will be of type A.

Consider the case study shown in Appendix 1 (a simplified version of BAPERS used in the previous Tutorials).

Question 1:

Consider the following numbers that characterize the operational profile for the simplified version of BAPERS.

1. The Receptionist takes on average 90 new jobs a day. During the day the receptionist would logs in on average 5 times (and would log out 5 times, of course).
2. All jobs taken in are processed and completed by the end of the day. Recording on BAPERS the processing of each job requires from a technician to do the following:
    a. Login,
    b. record the time spent on a job (i.e. ProcessJob use case) and
    c. Logout.
3. By the end of the day all jobs are collected by their respective customers (photographers who brought them in). The Receptionist would record the payment made by the respective customers. Each job is paid separately. No additional logins/logouts are needed for collecting payments. We assume that the Reception would stay logged-in most of the time, the 5 logins and logouts listed in point 1 above account for all logins and logouts by Receptionist during the day.

Build a sketch of the rationale that one might use in defining the ***operational profile*** for this simplified version of BAPERS. Compute the probabilities of each use-case run by any of the two actors. These probabilities together define the operational profile for BAPERS. Make sure that the sum of these probabilities is equal to 1.

Hint: Start the process by counting how many times each of the use cases shown in Appendix 1 can be 'run' by each of the actors – Receptionist and Technician during the day. Then you establish the total number of use cases executed during the day. Finally you establish the frequency with which each use case can be run by ANY actor. The probabilities in question are the ration of the frequencies over the total number of use cases.

Answer

The Receptionist can run 4 use cases: Login, Logout, TakeJob and Take payment. The Technician can only run Login, Logout and ProcessJob.

Using the description provided above we can produce the following table with the number of times each of the use cases is run by the actors:

	Login	*Logout*	*TakeJob*	*ProcessJob*	*TakePayment*	*Total*
Receptionist	*5*	*5*	*90*	*0*	*90*	*190*
Technician	*90*	*90*	*0*	*90*	*0*	*270*
Total	*95*	*95*	*90*	*90*	*90*	*460*

The total number of use cases executed by an actor (either Receptionist or Technician) is 460. Now we can easily compute the probability that a randomly chosen use cases is one of the 5 use cases possible with the simplified BAPPERS.

Login is run 95 times a day by both actors, the total number of use cases is 460. Hence:

P(Login) = 95/460 = 0.206522.

Similarly, Logout is run 95 times by both actors during the day, the total number of use cases is 460. Hence:

P(Logout) = 95/460 = 0.206522.

P(TakeJob) = 90/460 = 0.195652.

P(ProcessJob) = 90/460 = 0.195652.

P(TakePayment) = 90/460 = 0.195652.

Now we check whether the sum of these probabilities is 1, as it should be.

Question 2: Compute the profile that can be associated with each of the actors (Receptionist or Technician), e.g. count only the use cases that each of them can undertake during the day, the total number of use cases that each of them runs and compute the (conditional) probabilities of the respective use cases given that a randomly chosen use cases is run by the particular actor.

Answer

The difference is that instead of counting all use cases that can be run during the day by the actors we count only the use cases executed by either the Receptionist only or by the Technician only.

This is summarized in the table below:

	Login	*Logout*	*TakeJob*	*ProcessJob*	*TakePayment*	*Total*
Receptionist	*5*	*5*	*90*	*0*	*90*	*190*
Technician	*90*	*90*	*0*	*90*	*0*	*270*

As the total number of use cases run during the day by a Receptionist is 190 the conditional probabilities of the use cases given the actor is the Receptionist are as follows:

$P_R(Login) = 5/190 = 0.026316.$

$P_R(Logout) = 5/190 = 0.026316.$

$P_R(TakeJob) = 90/190 = 0.473684.$

$P_R(ProcessJob) = 0/190 = 0.$

$P_R(TakePayment) = 90/190 = 0.473684.$

Again the sum of these probabilities is 1.

For the Technician we use similar calculations: the Technician runs 270 use cases all together, hence:

$P_T(Login) = 90/270 = 0.33333.$

$P_T(Logout) = 90/270 = 0.33333.$

$P_T(TakeJob) = 0/190 = 0.$

$P_T(ProcessJob) = 90/270 = 0.33333.$

$P_T(TakePayment) = 0/270 = 0.$

Again the sum of these probabilities is 1 (rounding must be applied).

*Now you can appreciate that the **combined profile** that the two actors create for the system is very different from the profile each of them is using.*

Question 3 (complete at home especially if you have not installed jMeter on your laptop before ahead of tutorial): Conduct performance evaluation of example.org web-site using the tool jMeter. You can vary the number of "users" in the range [1 … 100]. Analyze the response times and the rate of error (i.e. the rate of responses which are returned with an error code that differ from 200, e.g. error code 502, 404, etc.). You may repeat the study with some other sites, e.g. Google.com or city.ac.uk.

Answer

No model answer is provided for this question. A demonstration of how to use jMeter was included in the lecture.

Part 2: Integration testing with jUnit

In this part of the tutorial we will use jUnit to conduct integration testing of a small subsystem derived from BAPPERS, which is shown in Appendix 2. It consists of an interface I_JobTask, implemented by the class JT_Implementation, which in turn is associated with some of the BAPPER classes that we have worked with in the previous tutorials. The diagram can be found under the "Implementation Model" of the vpp file released on Moodle.

Question 4:

Generate Java code from Visual Paradigm design class diagram (Implementation Model -> JobTask_CD). Create a Netbeans project for these files. Using the Netbeans support for jUnit, create a test class to test the methods listed in the I_JobTask interface. Write a setup Java code (@BeforeAll) in which you need to instantiate the implementation class and create several instances of TaksDescription: these instances must exist before a new Job can be created.

Answer

A minimalistic implementation in Java of the classes of the mini-BAPERS project are released on Moodle. These include:

- *I_JobTask.java*
- *Job.java*
- *JT_Implementation.java*
- *Task.java*
- *TaskDescription.java*

The test case, I_JobTaskTest.java, used for integration testing is also released on Moodle.

A video with a demonstration of how to create the integration test and how to run it in Netbeans IDE is also available on Moodle in the "Media Gallery".

Question 5:

Write tests for calculateJobPrice(jobID:int). This test should traverse an existing :Job object and take the sums of the prices of all Tasks, which belong to the particular :Job. Consider testing the following important cases:

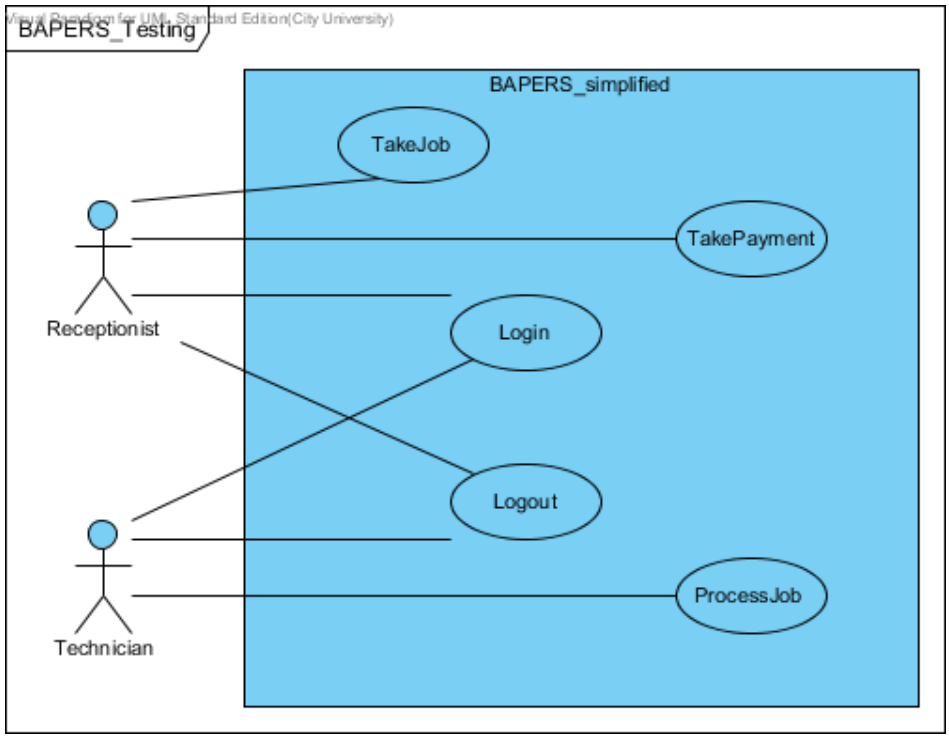
- jobID refers to an existing :Job object

- jobID refers to a non-existent :Job. Discuss with a colleague how you can detect this situation and the options to report to the jUnit the situation.

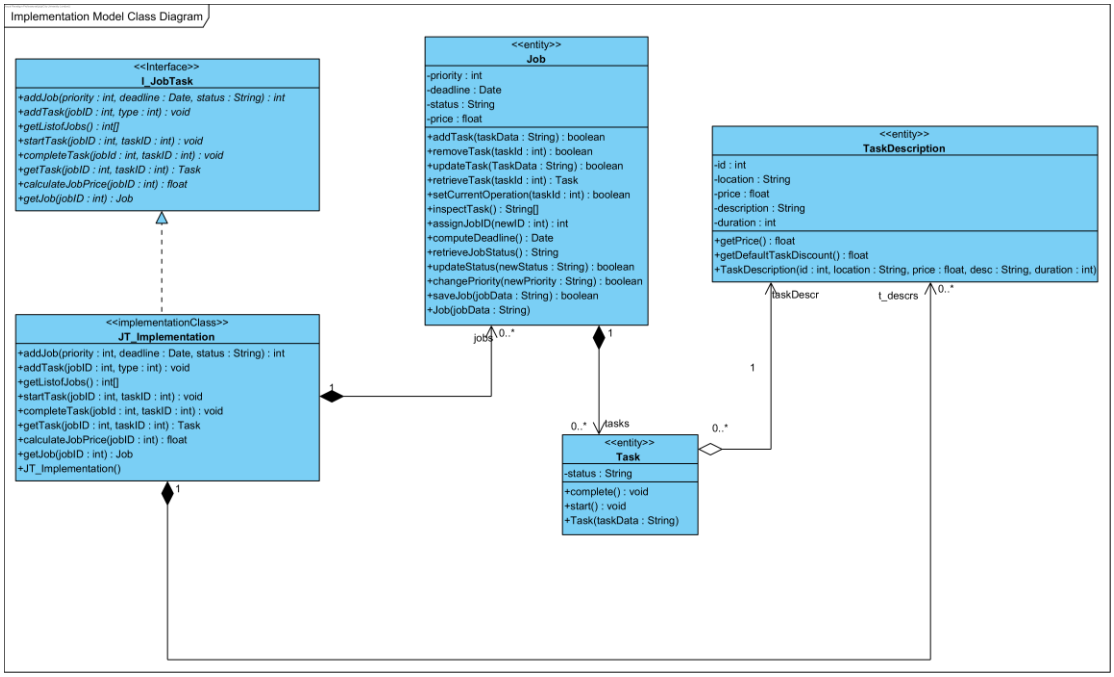
Answer

No model answer is provided for this test (the test does not provide any code for the method calculateJobPrice()).

APPENDIX 1



APPENDIX 2



Created: 3rd July 2013

Las updated: 27th of November 2018