

Model-Based Testing

(DIT848 / DAT260)

Spring 2013

Lecture 14 Revision

Gerardo Schneider

Department of Computer Science and Engineering
Chalmers | University of Gothenburg

Revision requests...

- Go through a previous exam
- A more complex example of ModelJUnit where the adapter is separated from the model

Exam MBT Disclaimer!

- Note that the following is only a sample of a previous exam!
- The precise content or format of the incoming exam might be slightly different!

Exam MBT (General issues)

- ALLOWED AID:
 - Books on testing
 - All lecture notes (including printouts of lectures' slides)
 - Students own notes
 - English dictionary
- NOT ALLOWED: Any form of electronic device (dictionaries, agendas, computers, mobile phones, etc)

Exam MBT (General issues)

- PLEASE OBSERVE THE FOLLOWING:
 - Motivate your answers (a simple statement of facts not answering the question is considered to be invalid);
 - Start each task on a new paper;
 - Sort the tasks in order before handing them in;
 - Write your student code on each page and put the number of the task on every paper;
 - Read carefully the section below "ABOUT THE FORMAT OF THE EXAM".

Exam MBT (General issues)

- ABOUT THE FORMAT OF THE EXAM:
 - The exam consists of 5 tasks, each one concerned with a specific part of the course content.
 - Each task is worth 20 points. In order to reach the level to pass with 3 (G) you need at least 50 points out of the total, and at least 6 points per task. To pass with 4 you need at least 65 points out of the total, and at least 8 points per task.
 - In order to pass with distinction (5/VG) you need to reach at least 80 points out of the total, and you must score at least 14 points per task.
- IMPORTANT: Note that you should have a minimum number of points per task in order to pass, so avoid letting unanswered tasks.

Exam MBT - May 21, 2012

- MBT-exam-2012-05-21.pdf

Task 1 - Test in general

Part 1

Solution

1. F - testing is always dynamic
2. T
3. F - debugging is testing + correcting the errors
4. F - This is the less advisable way to do it, according to many experts
5. F - No, you don't need a full implementation (you might use some mock code)

Task 1 - Test in general

Part 2

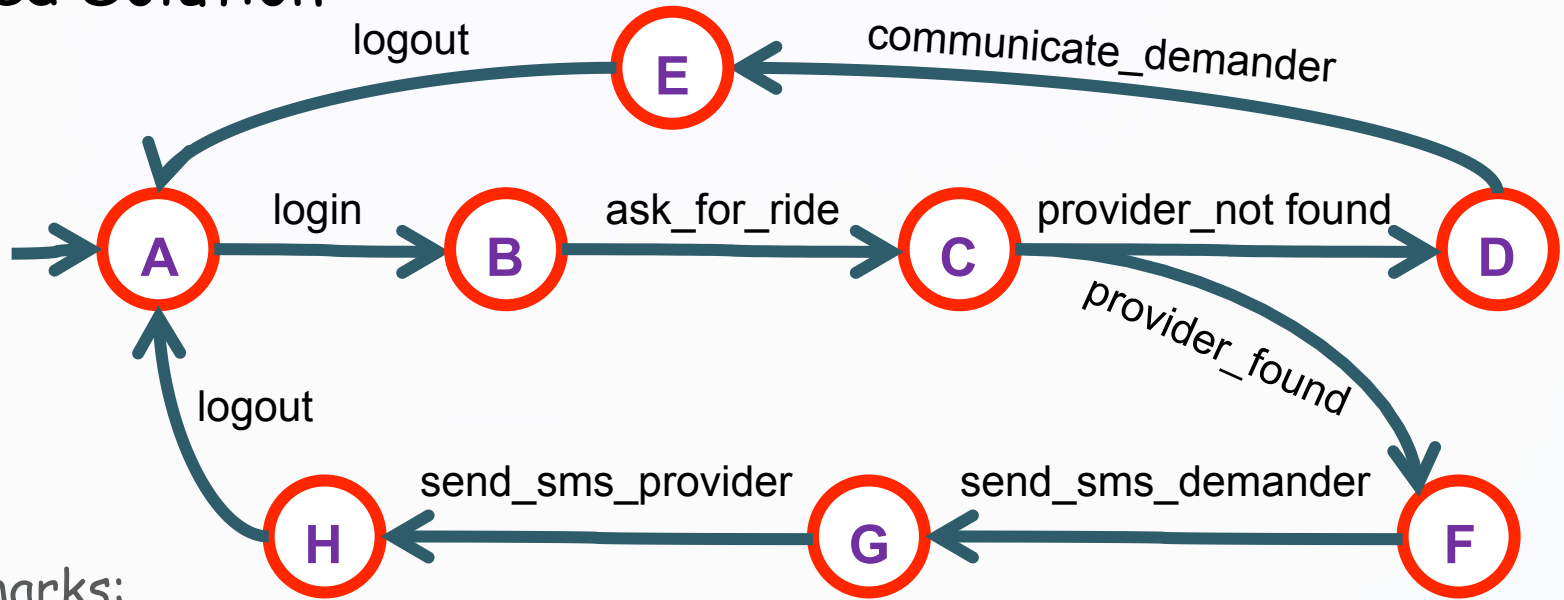
Solution:

1. Acceptance test (g) (also during system test - e)
2. stress/system test (e) and also acceptance (g)
3. Combination of coverage analysis (c) and unit tests (b)
4. timing response test (system test - e)
5. configuration test (system test - e)

Task 2 -State Machines

Part 1

Proposed Solution



Some remarks:

- Many other solutions depending on how much do you abstract
 - A "good" solution should be abstract enough as to capture the informal description (but not too much as to be useless)
- "logout" could be eliminated (as it is automatic)
- No check on whether login is correct or not (not in the specification)
- Implicit loop in state "C" on "look_for_provider"

Task 2 -State Machines

Part 2

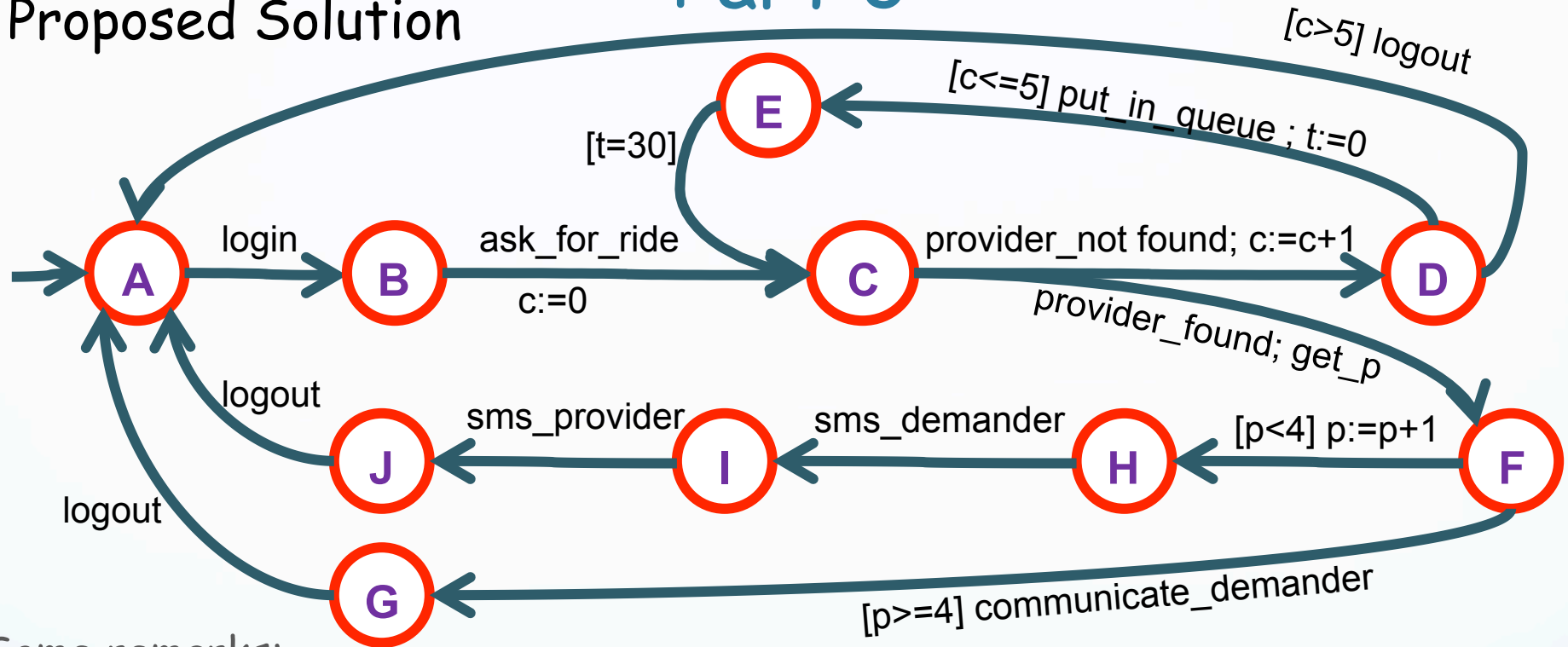
Proposed Solution

- Test cases you can extract:
 1. After login if there is provider then the demander gets an sms indicating that.
 2. If no provider exists for that ride then the user is logged out after getting a notification.
- Test cases you cannot extract:
 1. If a provider does exist for the ride, the user may still not get the guarantee of a ride due to overbooking.
 2. Any timing constraints in what concerns how much time to wait for getting a confirmation of a ride.

Task 2 - State Machines

Part 3

Proposed Solution



Some remarks:

- Brackets ("[]") are used as a short for "If ... then ..."
- t: timer; c: number of times a demander may request a ride; p: nr of passengers (stored in the DB; get using "get_p")
- Assumption: the timer is automatically incremented (implicit loop in state E)

Task 3 - White box testing and coverage

Part 1

Solution

- a. a-b-g
- b. (Considering the state as being between the transitions)
s1: d-a, d-e
s2: a-b, a-c
s3: c-d, g-d
s4: e-g, e-f, b-g, b-f, f-f, f-g
- c. e
a-b,
a-c-d-e,
e-g-d-e
a-b-g-d-e
- d. Add to the above
visiting "f" too
- e. a-b-g-d-e-f,
a-c-d-e

Task 3 - White box testing and coverage

Part 2

Solution

- a. Deterministic (i), initially connected (ii), minimal (iii), strongly connected (iv)
- b. Add copies of transitions a, g, d
(e.g: a-c-d-e-f-g-d'-a'-b-g'-d'')
- c. Transform the graph using de Bruijn's algorithm (dual graph) and then "Eulerize" it (see lecture 7)

Task 4 -MBT / ModelJUnit

Solution

1. F - you should aim at least at a 100% transition coverage
2. F - You might use transformation and adaptation.
3. F - you might need to change the code
4. F - this is the case for the transformation, not the adaptation
5. T
6. T
7. T
8. T
9. F - It doesn't as there might be many branches in the SUT abstracted away in the EFSM
10. F - Transition-based is control oriented, while pre/post is data-oriented.

Task 5 - Property-based test. and QuickCheck

Part 1

Solution

- a. $\text{prop_delete1 } x \ t =$
 $\text{delete } x \ (\text{delete } x \ t) == \text{delete } x \ t$
- b. $\text{prop_delete2 } x \ t = \text{not } (\text{member } x \ t) ==>$
 $\text{flatten } (\text{delete } x \ (\text{insert } x \ t)) == \text{flatten } t$
(Note that the it is not necessarily true that you get the same tree!)
- c. $\text{prop_delete3 } x \ t = (\text{member } x \ t) ==>$
 $(\text{flatten } (\text{insert } x \ (\text{delete } x \ t)) == \text{flatten } t)$
(Note that the it is not necessarily true that you get the same tree!)
- d. (The statement should be read as "Write a property that checks that 2 BSTs are not equal if they don't contain the same elements.")
 $\text{prop_equal } t1 \ t2 =$
 $\text{not } (\text{flatten } t1 == \text{flatten } t2) ==> t1 \ /= \ t2$

* Thanks to Grégoire Détrez for finding and fixing some bugs on a previous solution

20 min

Task 5 - Property-based test. and QuickCheck

Part 2

Solution

- a. F - you write properties, not necessarily a full model.
- b. T
- c. F - There is no guarantee of getting the same tree. You should write:

```
prop_merge1 x y t1 t2 = flatten (merge (insert x t1)
(insert y t2)) == flatten (insert x (insert y (merge t1 t2)))
```
- d. F - The problem is that the symbols < and > are interchanged. You should make the following change:
"&& all (<y) (flatten lt) && all (>y) (flatten rt)"

2nd request...

- An example on how to write a model and (separated) adapter of a calculator may be found here (by your colleague Erik Lindqvist):

[CalcAdapter.java](#)

[CalcFsm.java](#)

* Thanks to Erik for sharing his solution!

Exam

- **June 1st, at 08:30 (Lindholmen)**
 - Note time was changed (by Study Administration)