DECISIONS, DECISIONS – HOW TO USE DECISION TABLES FOR EFFECTIVE TEST DESIGN

RANDALL W. RICE, CTAL-FULL, CTFL-AT, CMT
WWW.RICECONSULTING.COM

© 2016, Rice Consulting Services, Inc.

AGENDA

• The main concept of decision tables
• How to create simple decision tables
• How to create decision tables with invalid dependencies
• Why and how to reduce decision tables
• How decision tables help in test design
• How decision tables help in requirements analysis
• Tool support for creating decision tables
• Explanation of ASTQB Foundation Level Sample Question #24
• Additional Resources
DECISION TABLE TESTING

- Decision tables are a good way to capture system requirements that contain logical conditions, and to document internal system design.
- They may be used to record complex business rules that a system is to implement.
- Can be applied as part of both black-box and white-box test design techniques.

KEY POINT

- You do not need complete definition of a requirement to apply decision tables.
- That is the beauty of the technique.
  - Tell me one rule and I can deduce the rest.
- That's why this is also great for requirements analysis!
  - How many times have you seen "gaps" in requirements where certain combinations of conditions or events were not considered?
**DECISION TABLE TESTING (2)**

- A logical way to derive test cases
- Best applied with a limited number of rules
  - (7 rules with T/F decisions yields 128 possible test cases. 8 rules – 256 cases, etc.)
  - The formula is $2^n$, where $n$ is the number of conditions
- Each rule can be seen as:
  - T : True
  - F : False
  - I : (Invalid, illogical)

**“SIMPLE” EXAMPLE**

- If an employee is hourly and works over 40 hours in a weekly pay period, they are paid overtime for each hour worked over 40 hours. Overtime pay is 1.5 times the regular hourly pay.

Step 2: Define the T/F (or Y/N) values. Tip: Do the “happy path” first, then vary the values.

<table>
<thead>
<tr>
<th>Conditions</th>
<th>Rule 1</th>
<th>Rule 2</th>
<th>Rule 3</th>
<th>Rule 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee is hourly</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>Over 40 hours worked in a week</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Actions</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Overtime pay (1.5 x) for each hour worked &gt; 40</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No overtime pay</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
</tr>
</tbody>
</table>
IDENTIFYING ILLOGICAL CONDITIONS

- Notice that in Rules 2, 3 and 4, the outcome is the same – No overtime paid.
- However, in Rules 3 and 4, we really don’t care if the person works over 40 hours or not. They aren’t eligible for OT anyway.

<table>
<thead>
<tr>
<th>Conditions</th>
<th>Rule 1</th>
<th>Rule 2</th>
<th>Rule 3</th>
<th>Rule 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee is hourly</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>Over 40 hours worked in a week</td>
<td>T</td>
<td>F</td>
<td>I</td>
<td>I</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Actions</th>
<th>Rule 1</th>
<th>Rule 2</th>
<th>Rule 3</th>
<th>Rule 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overtime pay (1.5 x) for each hour worked &gt; 40.</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>No overtime pay</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

REDUCING THE TABLE

- We can consider the condition of working over 40 hours in a week invalid or illogical if the employee is not hourly.
- Now, notice that Rules 3 and 4 are identical, both in conditions and outcome.
- So we only need 3 Rules or test cases for full decision table coverage.
- However, you may choose to perform all 4 cases with T/F values only as negative tests for cases 3 and 4.
WHY REDUCE A TABLE?

• To achieve more efficiency in test design.
  • Fewer cases, no loss in condition coverage.
• However, as we just mentioned you may want to test all combinations of conditions, including the “invalid” ones, as negative tests.
  • There may be cases where the invalid cases are impossible to achieve.
  • Example: Testing the IE browser on a Mac O/S.

MORE COMPLEX DECISION TABLE EXAMPLE

• Rule 1 to be tested says:
  • If all of the following are true:
    • Employee is hourly
    • Hours worked in a pay period is over 40
    • No sick time, vacation or holiday time accrued during the week in excess of OT
    • No holidays worked during the week
  • Then the following actions take place
    • Pay is computed as 40 hours at the employee’s regular pay rate
    • Overtime pay for each hour over 40
    • Overtime hourly rate is 1.5 times employee’s regular rate.
### DECISION TABLE EXAMPLE (2)

<table>
<thead>
<tr>
<th>Condition</th>
<th>Rule 1</th>
<th>Rule 2</th>
<th>Rule 3</th>
<th>Rule 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee is hourly</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>Hours worked in a pay period is over 40</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>Sick time, vacation or holiday taken &lt; OT hours</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>I</td>
</tr>
<tr>
<td>No holidays worked during week</td>
<td>T</td>
<td>F</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>Actions</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hours up to 40 computed at regular pay,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40 hours over, less sick, vacation and holiday</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>hours, computed at 1.5 times regular rate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Holiday pay computed for hours worked on the holiday at 2 times regular rate</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No overtime pay rate applied</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

### HOW DECISION TABLES HELP IN TEST CASE DESIGN

- They identify logical variations
- They identify variations that do not need to be tested because of illogical combinations
- They help determine test case completeness
HOW MUCH TO COVER?

• The coverage standard commonly used with decision table testing is to have at least one test per column, which typically involves covering all combinations of triggering conditions.

WHITE-BOX EXAMPLE – MULTIPLE CONDITION COVERAGE

IF (A > B) or ((C < D) and (A < B)) THEN
PERFORM X
ELSE
PERFORM Y;

<table>
<thead>
<tr>
<th>A&gt;B</th>
<th>C&lt;D</th>
<th>A&gt;B</th>
<th>O/C</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>X</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>X</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>X</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>X</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>X</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>Y</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Y</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>Y</td>
</tr>
</tbody>
</table>

1. Create decision table
2. Eliminate invalid or duplicate cases
3. Convert to test cases

Note the test cases are rows in this example instead of columns.
HOW DECISION TABLES ARE CREATED

- Business Rules
- Functional Requirements
- Use Cases
- Processes
- Test Designer
- Application Functions
- Code
- Decision Table

TOOLS

- This is often a manual effort
- However, a great tool for automating the work is Richard Bender’s BenderRBT tool.
  - http://www.benderrbt.com
FINALLY...INFAMOUS QUESTION
#24 FROM THE ASTQB SAMPLE EXAM

#24 You have been given the following conditions and results from those condition combinations. You can only have one form of payment. A PIN is only needed for a debit card. Given this information, using the decision table technique, what is the minimum number of test cases you would need to test these conditions?

<table>
<thead>
<tr>
<th>Conditions:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid cash</td>
</tr>
<tr>
<td>Valid credit card</td>
</tr>
<tr>
<td>Valid debit card</td>
</tr>
<tr>
<td>Valid PIN</td>
</tr>
<tr>
<td>Bank accepts</td>
</tr>
<tr>
<td>Valid Selection</td>
</tr>
<tr>
<td>Item in Stock</td>
</tr>
<tr>
<td>Results:</td>
</tr>
<tr>
<td>Reject Cash</td>
</tr>
<tr>
<td>Reject Card</td>
</tr>
<tr>
<td>Error Message</td>
</tr>
<tr>
<td>Return Cash</td>
</tr>
<tr>
<td>Refund Card</td>
</tr>
<tr>
<td>Sell Item</td>
</tr>
</tbody>
</table>

1. 7
2. 13
3. 15
4. 18

MY SOLUTION APPROACH

For the purposes of answering the exam question, I do not spend time on defining the results.
• The value of decision table testing is that it creates combinations of conditions that might not otherwise have been identified during test design or exercised during testing.

• It may be applied to all situations when the action of the software depends on several logical decisions.
FURTHER REFERENCES

• The Art of Software Testing by Glenford Myers
  • He ties together decision tables with cause-effect graphing.
• Software Testing Techniques, 2nd Ed. By Boris Beizer
  • Very complete treatment of the topic.
• www.abebooks.com
• www.alibris.com

YOUR QUESTIONS?
BIO - RANDALL W. RICE

- Over 35 years experience in building and testing information systems in a variety of industries and technical environments
- ISTQB Certified Tester – Foundation level (CTFL), Advanced Level (CTAL) Full
- ASTQB Certified Mobile Tester (CMT)
- ISTQB Foundation Level Agile Tester (CTFL-AT)
- Director, American Software Testing Qualification Board (ASTQB)
- Chairperson, 1995 - 2000 QAI’s annual software testing conference
- Principal Consultant and Trainer, Rice Consulting Services, Inc.
CONTACT INFORMATION

Randall W. Rice, CTAL
Rice Consulting Services, Inc.
1608 SW 113th Pl
Oklahoma City, OK  73170
Ph: 405-691-8075
Fax: 405-691-1441
Web site: www.riceconsulting.com
e-mail: rrice@riceconsulting.com