Advanced Software Testing

Applying Decision Tables to Business Logic

RBCS

TIME TESTED.
TESTING IMPROVED.

www.RBCS-US.com

Advanced Software Testing

- A series of webinars excerpted from *Advanced Software Testing: Volume 1,* a book for test analysts and test engineers
- Equivalence partitioning and boundary value analysis are useful for testing input field validation
- Three additional techniques are handier and more powerful for business logic
  - Decision tables
  - State based testing
  - Use cases
- This first webinar covers the related concepts of decision tables and cause-effect graphs

Decision Tables

- Concept: test the rules that govern handling of transactional situations
- Model: table (or Boolean graph) connecting conditions with actions
- Test derivation: fulfill conditions, check actions
- Coverage criteria: at least one test per combination of conditions (DT column)
- Bug hypothesis: improper action or missing action

Example: Decision Table (Full)

Conditions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Real account?	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N
Active account?	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	N	N	N	N
Within limit?	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N	Y	Y	N	N
Location okay?	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N
Actions																
Approve?	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Call cardholder?	N	Y	Y	Y	N	Y	Y	Y	N	N	N	N	N	N	N	N
Call vendor?	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

Example: Deriving Tests

- In the example just shown, each column of the table is a test case
  - We will create the conditions (which are the test's inputs)
  - We will verify the actions (which are the test's expected results)
- In some cases, we might generate more than one test case per column (more later)
- In this case, some of the test cases don't make much sense; e.g.:
  - Account not real but account active?
  - Account not real but account within limit?
- Maybe we don't need all the columns in our decision table?

Collapsing a Decision Table

- If the value of one or more particular conditions can't affect the actions for two or more combinations of conditions, we can collapse the decision table
- This involves combining two or more columns
  - Combinable columns often **but not always** next to each other
  - Look for two or more columns that result in the same combination of actions (for all the actions in the table)
  - Replace the conditions that are different in those columns with "-" (for don't care/doesn't matter/can't happen)
- Repeat this process until no further columns share the same combination of actions or where collapse would erase an important distinction
- Be careful with tables that have non-exclusive rules

Example: Decision Table (Collapsed)

Conditions	1	2	3	5	6	7	9
Real account?	Y	Y	Y	Y	Y	Y	N
Active account?	Y	Y	Y	N	N	N	-
Within limit?	Y	Y	N	Y	Y	N	-
Location okay?	Y	N	-	Y	N	-	-
Actions							
Approve?	Y	N	N	N	N	N	N
Call cardholder?	N	Y	Y	N	Y	Y	N
Call vendor?	N	N	N	Y	Y	Y	Y

Column numbers retained for ease of reference to full table

Study carefully to understand why rule 4 could collapse into rule 3, but not rule 3 into rule 2

The same logic also applies to rule 8 collapsing into rule 7, but not rule 7 into rule 6

Formula for number of columns ($2^{conditions}$) no longer applies

Regular pattern of conditions no longer applies

Converting to/from a Cause-Effect Graph

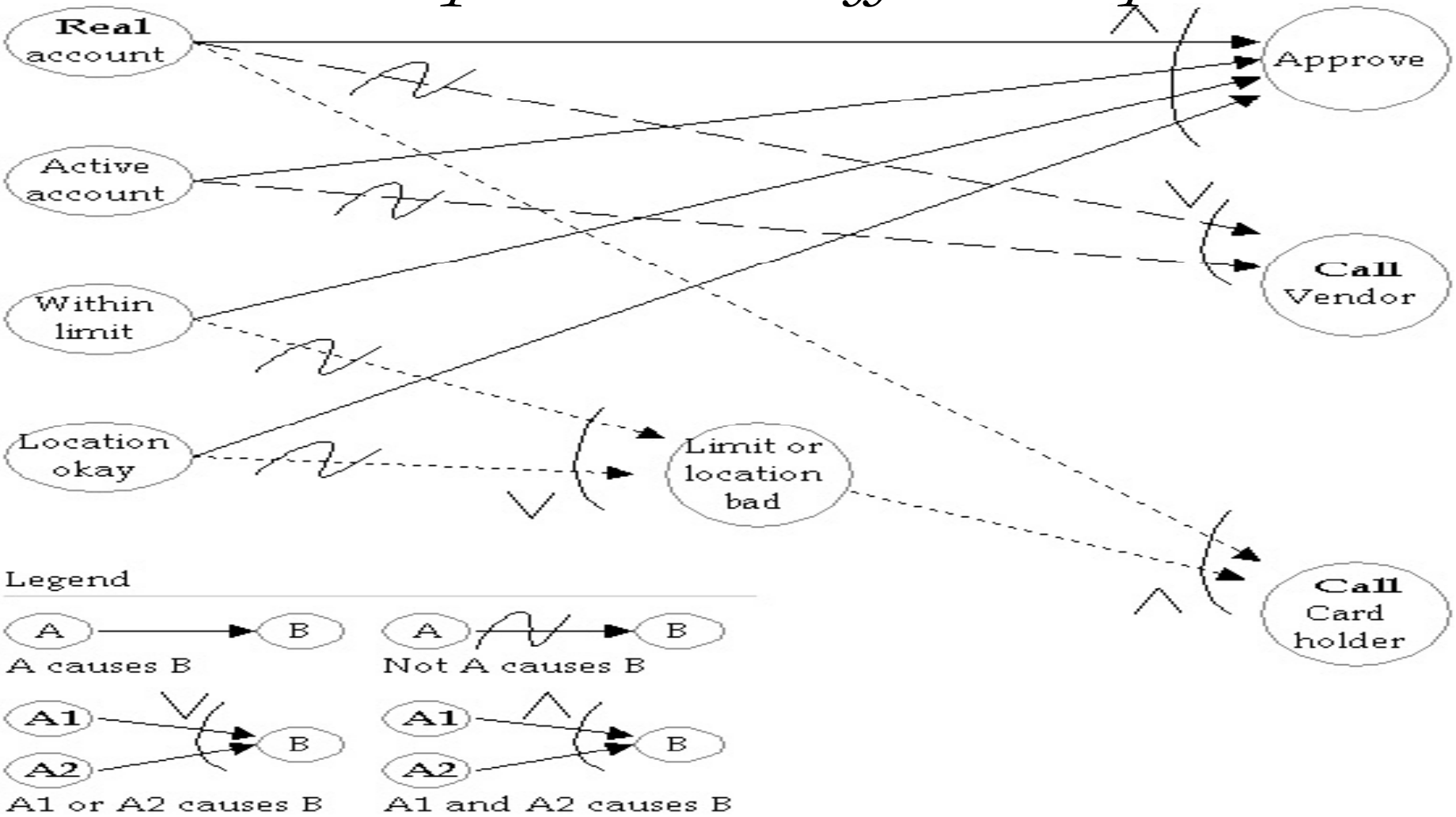
Table to graph

- List all the conditions on left of a blank page
- List all the actions on right of a blank page
- Read the table to identify how combinations of conditions cause an action
- Connect one or more conditions with each action using Boolean operators
- Repeat for all actions

Graph to table

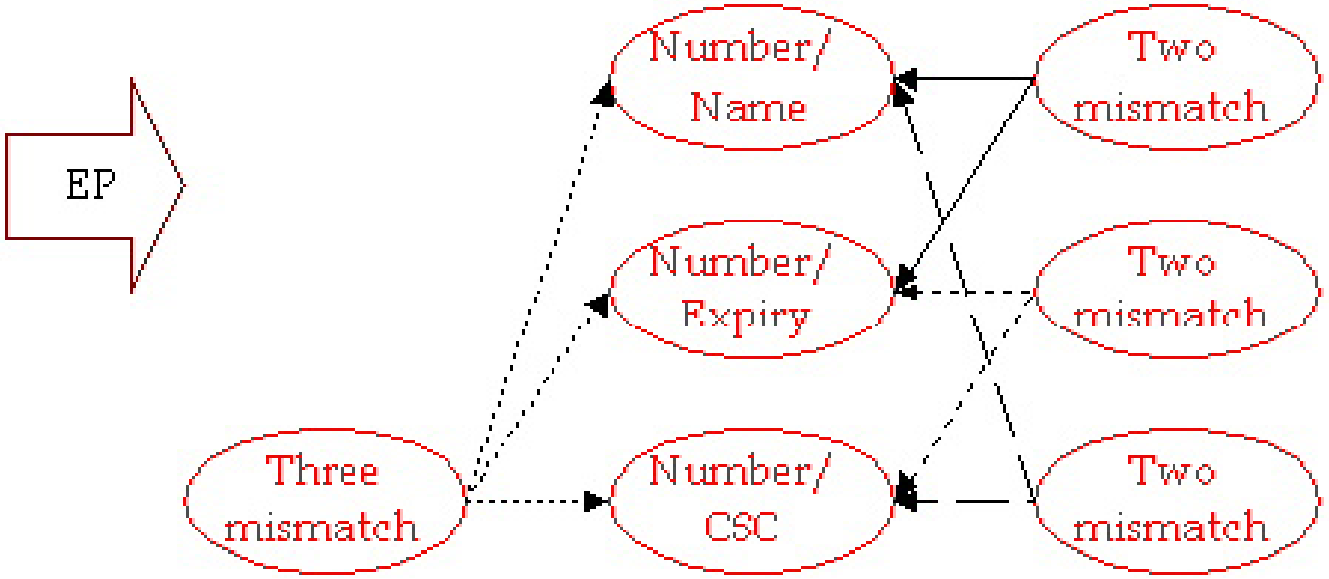
- List all the conditions on the top left of decision table
- List all the actions on the bottom left of decision table
- Generate all possible combinations of conditions
- Determine actions taken/not taken for each combination using graph
- Collapse when complete if desired

Example: Cause-Effect Graph



Equivalence Partitions and Decision Tables

Conditions	9
Real account?	N
Active account?	-
Within limit?	-
Location okay?	-



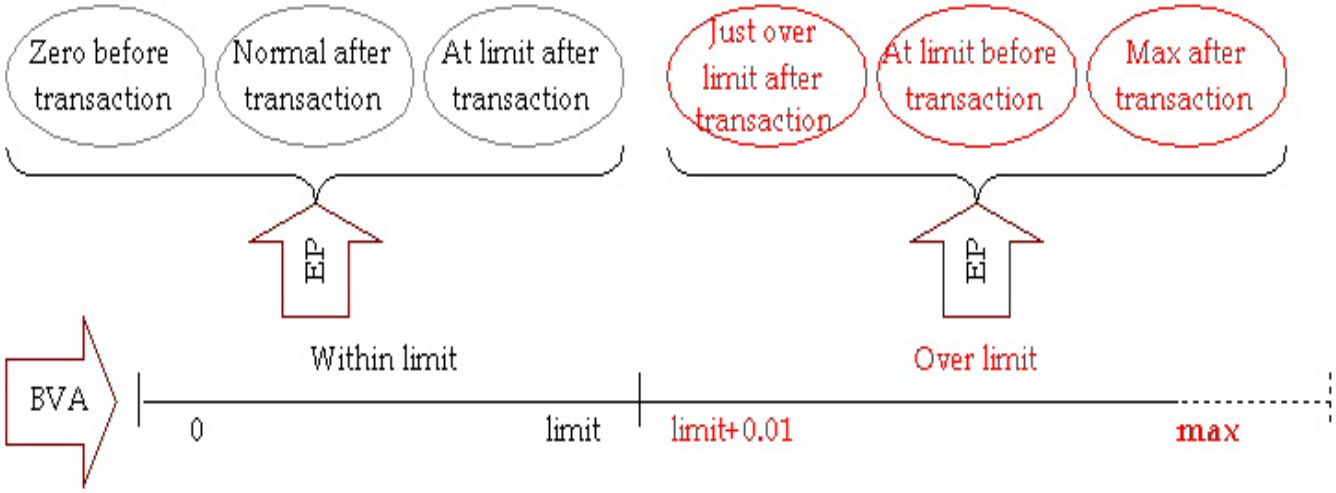
To find interesting tests for this column, apply equivalence partitioning:

- Card number and cardholder mismatch
- Card number and expiry mismatch
- Card number and CSC mismatch
- Two of the above mismatches (three possibilities)
- All three mismatches

So, there could be seven tests for that column

Boundary Values and Decision Tables

Conditions	1	2	3	5	6	7
Real account?	Y	Y	Y	Y	Y	Y
Active account?	Y	Y	Y	N	N	N
Within limit?	Y	Y	N	Y	Y	N
Location okay?	Y	N	-	Y	N	-



To find interesting tests for within limit, apply boundary value analysis :

- The account starts at zero balance
- The account would be at a normal balance after transaction
- The account would be exactly at the limit after the transaction
- The account would be exactly over the limit after the transaction
- The account was at exactly the limit before the transaction
- The account would be at the maximum overlimit value after the transaction

So, there would be four within-limit tests and three over-limit tests

Example: Non-exclusive Rules

- Sometimes more than one rule can apply to a transaction
- This complicates testing somewhat
- First, test each by itself, with no conditions not related to that rule met
- Next, consider testing combinations
  - Avoid combinatorial explosions
  - Use risk to weight combinations

Conditions	1	2	3
Foreign exchange?	Y	-	-
Balance forward?	-	Y	-
Late payment?	-	-	Y
Actions			
Exchange fee?	Y	-	-
Charge interest?	-	Y	-
Charge late fee?	-	-	Y

Conclusion

- We've seen how to apply decision tables and cause effect graphs to testing
- Useful for sophisticated and complex internal business logic in applications
- Decision tables are a great way to test detailed business rules in isolation, especially for transactional types of situations
- In the next two webinars, we'll look at two additional techniques, use cases and state-based test techniques