Advanced Software Testing

Applying State Diagrams 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 second webinar covers state based testing

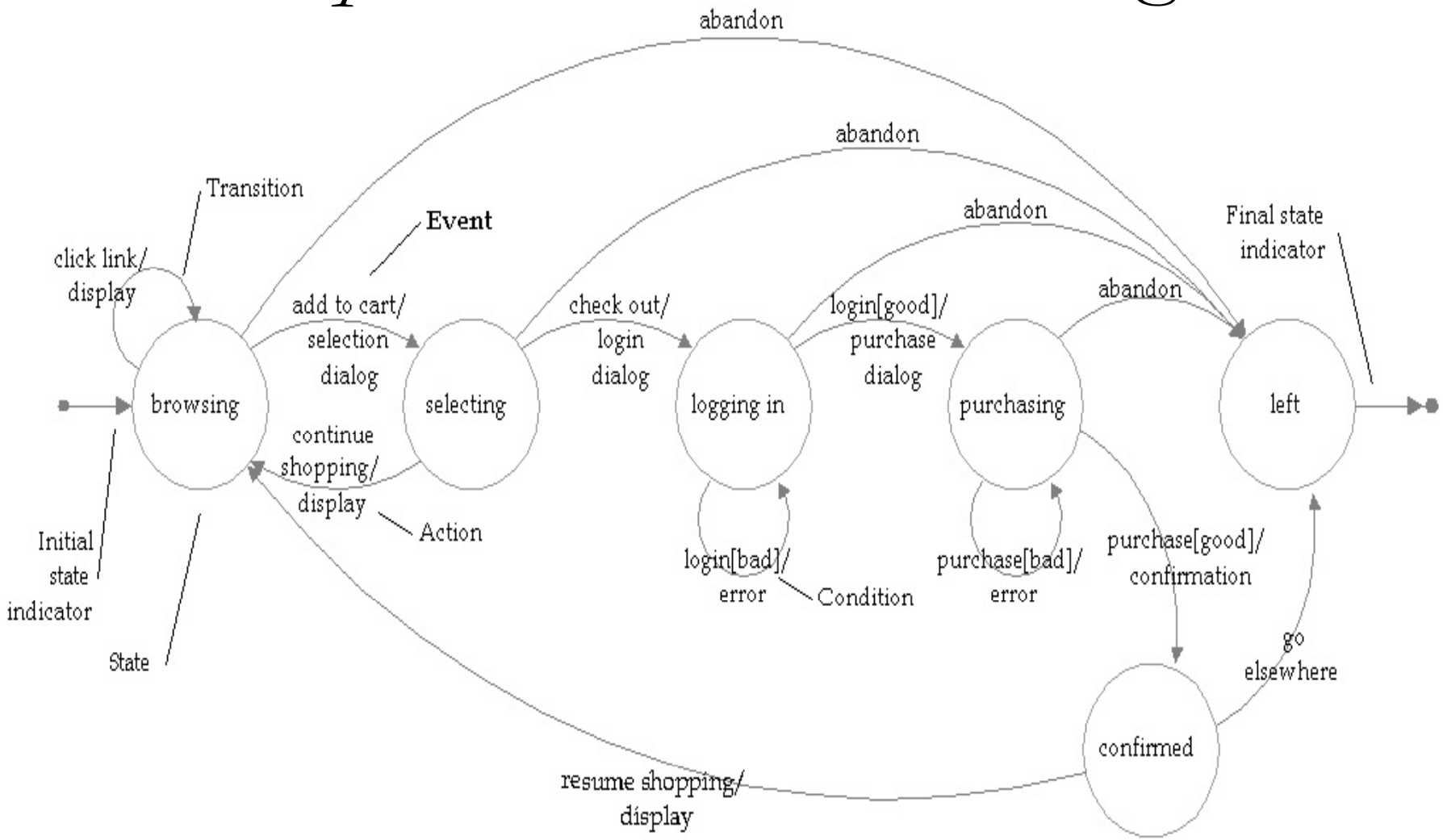
Copyright (c) 2008-2010 RBCS

State-Based Testing

- Concept: test handling of (arbitrarily-long) sequences of events
- Model: state transition diagram or table connecting beginning states, events, and conditions with resulting states and actions
- Test derivation: traverse the states and transitions, check the states and actions
- Coverage criteria: a) all states, transitions; b) all rows; or, c) transition sequences (*N* length)
- Bug hypothesis: wrong action or wrong transition given event history and conditions

Copyright (c) RBCS 2008-2010

Example: State Transition Diagram



Copyright (c) RBCS 2008-2010

Deriving Tests (State/Transition Cover)

1. Adopt a rule for where a test must start and where it may or must end (e.g., "Test must start in an initial state and may only end in a final state")
2. From an allowed test starting state, define a sequence of event/condition combinations that leads to an allowed test ending state, capturing the expected action for each transition
3. As you visit each state and traverse each transition, mark it as covered (e.g., using a marker on a printed version)
4. Repeat steps 2 and 3 until all states have been visited and all transitions traversed

www.rbcs-us.com
Copyright (c) RBCS 2008-2010

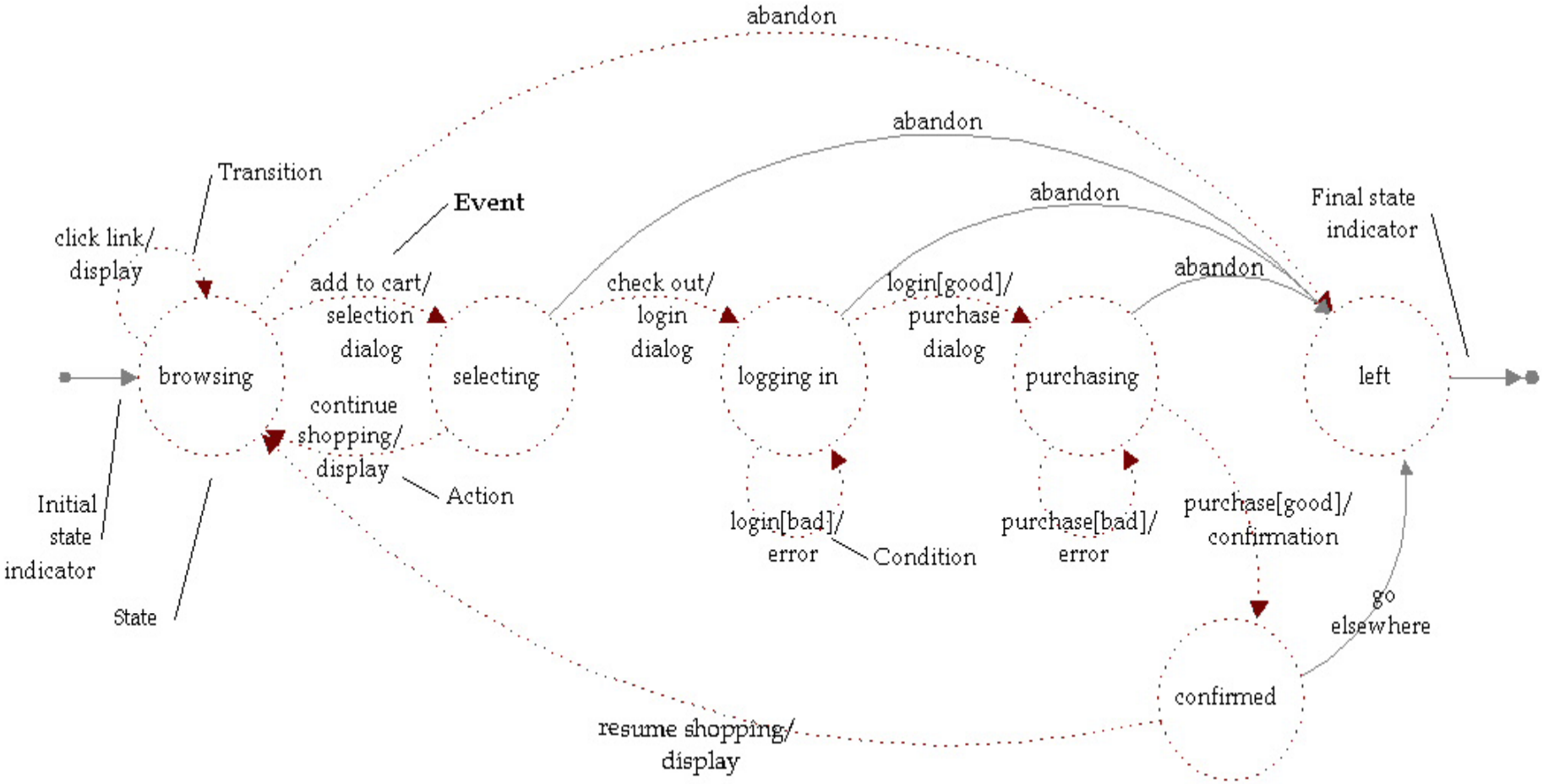
Example: Deriving Tests 1

- Rule: A test must start in the initial state and must end in the final state

1. Generate the first test: (browsing, click link, display, add to cart, selection dialog, continue shopping, display, add to cart, selection dialog, checkout, login dialog, login[bad], error, login[good], purchase dialog, purchase[bad], error, purchase[good], confirmation, resume shopping, display, abandon, left)
   Check completeness of coverage

Copyright (c) RBCS 2008-2010

Coverage Check 1



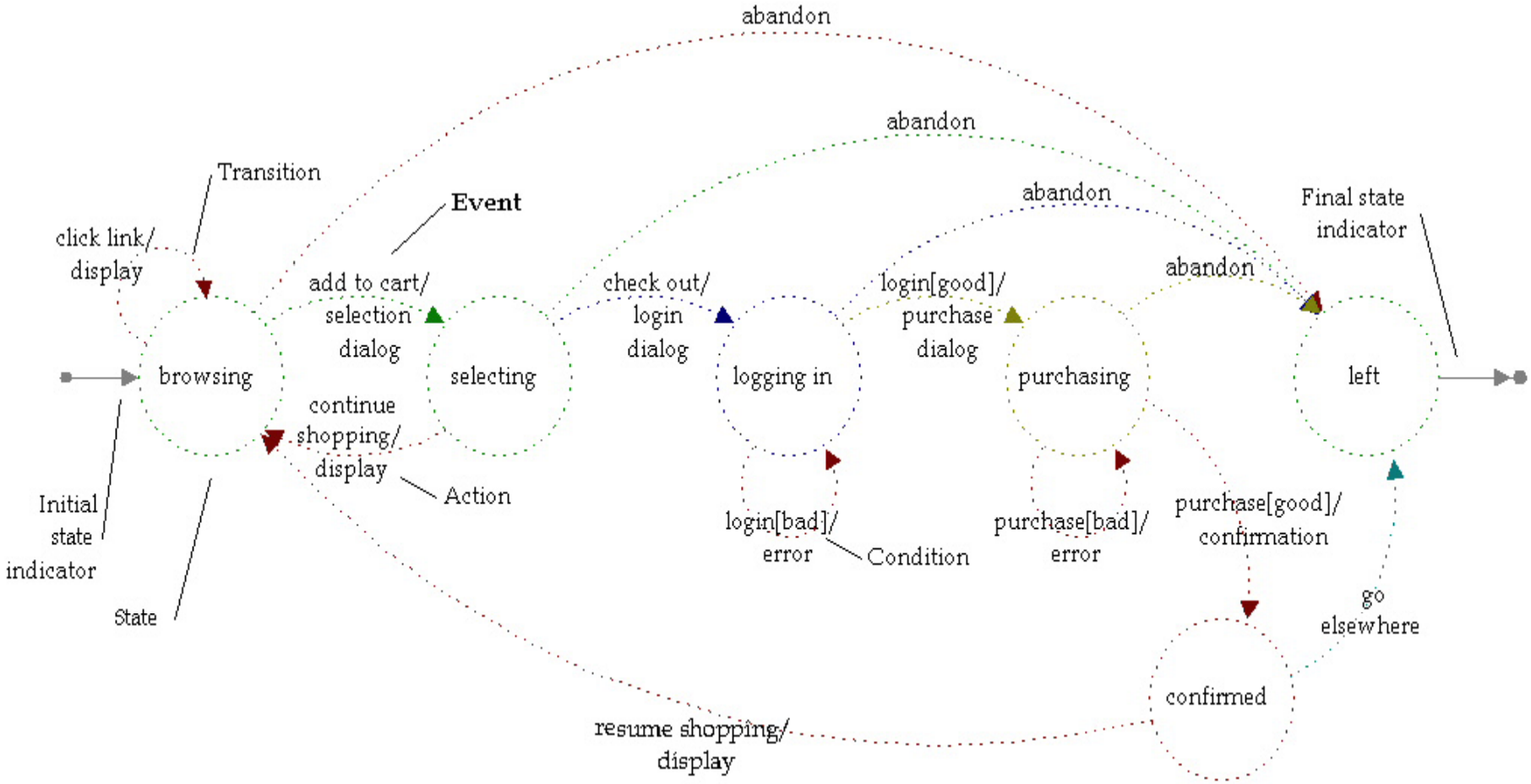
Copyright (c) RBCS 2008-2010

Example: Deriving Tests 2

1. (browsing, click link, display, add to cart, selection dialog, continue shopping, display, add to cart, selection dialog, checkout, login dialog, login[bad], error, login[good], purchase dialog, purchase[bad], error, purchase[good], confirmation, resume shopping, display, abandon, left)
2. (browsing, add to cart, selection dialog, abandon, <no action>, left)
3. (browsing, add to cart, selection dialog, checkout, login dialog, abandon, <no action>, left)
4. (browsing, add to cart, selection dialog, checkout, login dialog, login[good], purchase dialog, abandon, <no action>, left)
5. (browsing, add to cart, selection dialog, continue shopping, display, add to cart, selection dialog, checkout, login dialog, login[good], purchase dialog, purchase[good], confirmation, go elsewhere, <no action>, left)

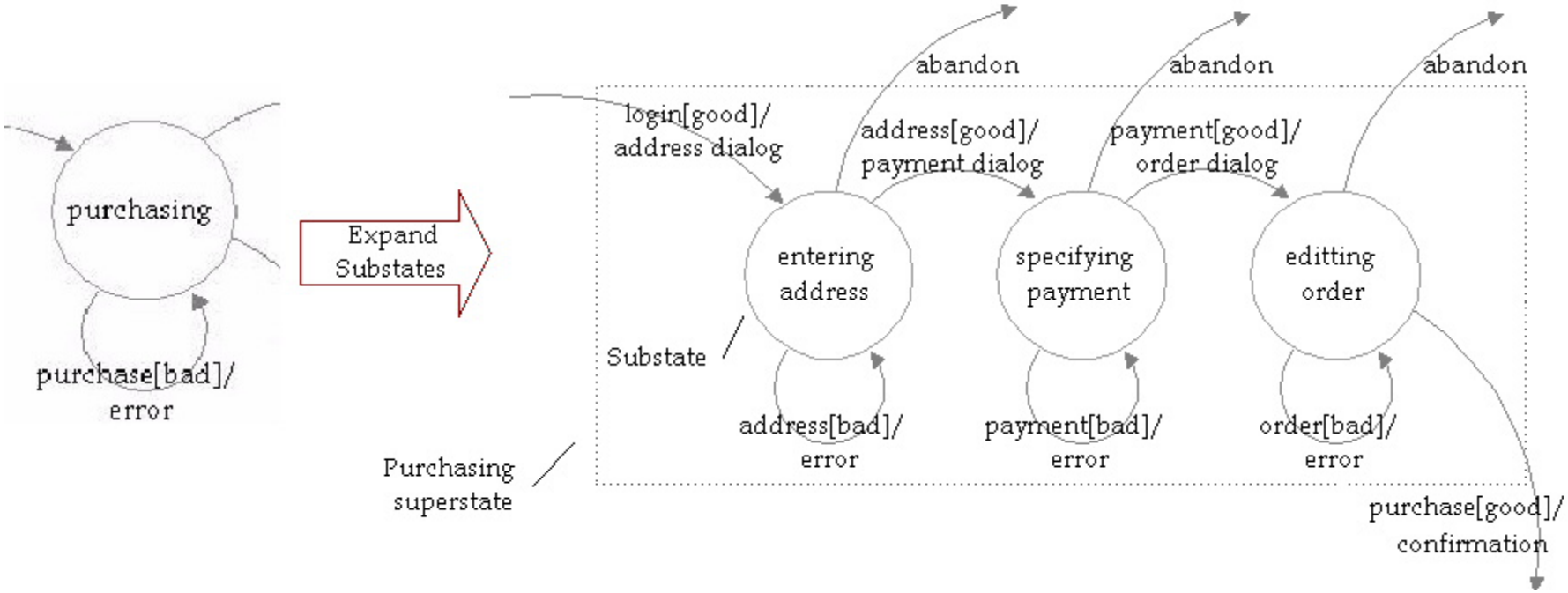
Copyright (c) RBCS 2008-2010

Coverage Check 2



www.rbcs-us.com
Copyright (c) RBCS 2008-2010

Example: Superstates and Substates



In some cases, it makes sense to expand a single state into a superstate consisting of two or more substates

Cover all transitions into the superstate, all transitions out of the superstate, all substates, and all transitions within the superstate

Copyright (c) RBCS 2008-2010

Example: State Transition Table

Browsing
Selecting
Logging in
Purchasing
Confirmed
Left

×

Click link
Add to cart
Continue shopping
Check out
Login[bad]
Login[good]
Purchase[bad]
Purchase[good]
Abandon
Resume shopping
Go elsewhere

=

Current State	Event/cond	Action	New State
Browsing	Click link	Display	Browsing
Browsing	Add to cart	Selection dia	Selecting
Browsing	Continue shopping	Undefined	Undefined
Browsing	Check out	Undefined	Undefined
Browsing	Login[bad]	Undefined	Undefined
Browsing	Login[good]	Undefined	Undefined
Browsing	Purchase[bad]	Undefined	Undefined
Browsing	Purchase[good]	Undefined	Undefined
Browsing	Abandon	<no action>	Left
Browsing	Resume shopping	Undefined	Undefined
Browsing	Go elsewhere	Undefined	Undefined
Selecting	Click link	Undefined	Undefined
{Fifty-three rows, generated in the pattern shown above, not shown}			
Left	Go elsewhere	Undefined	Undefined

State transition tables force consideration of pairs of states with event/condition combinations that might be forgotten

To construct, list all states and all event/condition combinations. Create a table with a row for each pair of state and event/condition combination, each row with four fields: current state; event/condition; action; and, new State. Populate action and new state based on the state transition diagram; the other rows are undefined

Copyright (c) RBCS 2008-2010

Deriving Tests (Row Cover)

1. Start with a set of tests (including the start/stop rule), derived from a state transition diagram, that achieves state/transition cover
2. Construct the state transition table and confirm that the tests cover all the defined rows (they should)
3. Select a test that visits a state for which one or more undefined rows exists in the table, and modify that test to attempt to introduce the undefined event/condition combination for that state
4. As you modify the tests, mark the row as covered (e.g., using a marker on a printed version)
5. Repeat steps 3 and 4 until all rows have been covered

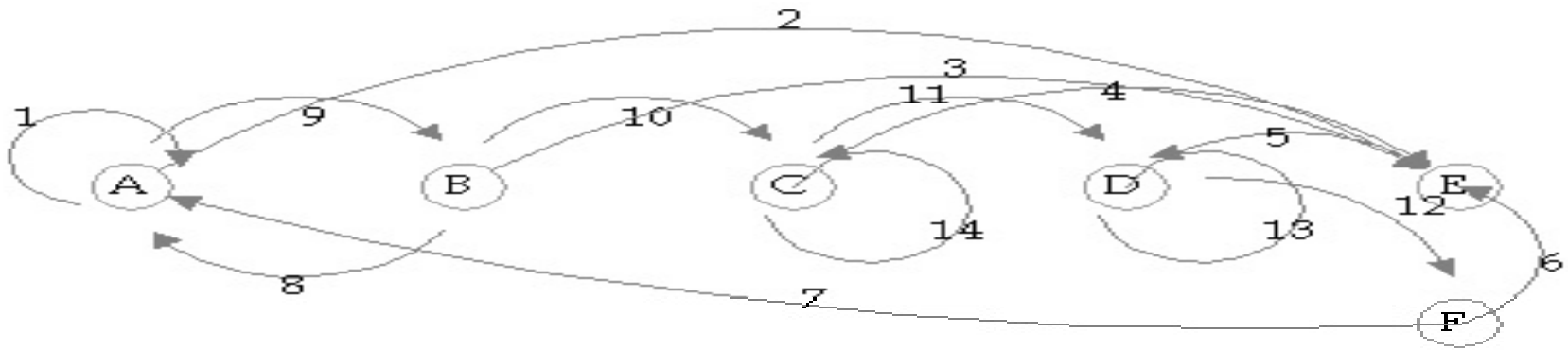
Copyright (c) RBCS 2008-2010

Example: Deriving Tests

- Existing test
  - (browsing, add to cart, selection dialog, checkout, login dialog, login[good], purchase dialog, abandon, <no action>, left)
- Modified tests (to cover undefined browsing event/conditions only)
  - (browsing, *attempt:* continue shopping, *action undefined*, add to cart, selection dialog, checkout, login dialog, login[good], purchase dialog, abandon, <no action>, left)
  - (browsing, *attempt:* check out, *action undefined*, add to cart, selection dialog, checkout, login dialog, login[good], purchase dialog, abandon, <no action>, left)
  - There are six other modified tests for browsing, not shown…
- Don't try to cover undefined event/conditions combinations for more than one state in any test, because you don't know whether the system will remain testable!
- Best case scenario is that the undefined event/condition combination is ignored or rejected with an intelligent error message, and processing continues normally from there

Copyright (c) RBCS 2008-2010

Example: N-1 Switch Coverage



0-switch			1-switch								
A1	A2	A9	A1A1	A1A2	A1A9				A9B10	A9B8	A9B3
B10	B8	B3	B10C14	B10C11	B10C4	B8A1	B8A2	B8A9			
C14	C11	C4	C14C14	C14C11	C14C4	C11D13	C11D12	C11D5			
D13	D12	D5	D13D13	D13D12	D13D5	D12F6	D12F7				
F6	F7					F7A1	F7A2	F7A9			

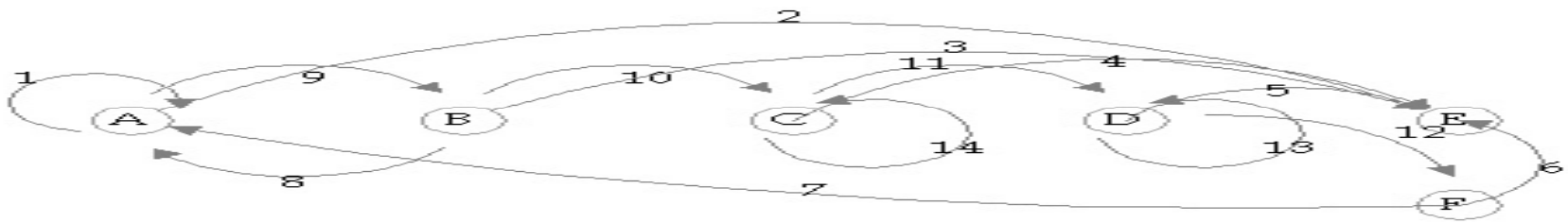
Copyright (c) RBCS 2008-2010

Deriving Tests (N-1 Switch Cover)

1. Start with a set of tests (including the start/stop rule), derived from a state transition diagram, that achieves state/transition cover
2. Construct the switch table, confirm that the tests cover 0-switch columns (they should), then check for higher-order switches covered by the tests
3. Using 0-switch sequences as needed, construct a test that reaches a state from which an uncovered higher-order switch originates, include that switch in the test, and repeat until the test must terminate
4. As you construct tests, mark switches as covered (e.g., using a marker on a printed version)
5. Repeat steps 3 and 4 until all switches have been covered

Copyright (c) RBCS 2008-2010

Example: Deriving Tests



0-switch			*1-switch*								
A1	A2	A9	A1A1	A1A2	A1A9				A9B10	A9B8	A9B3
B10	B8	B3	B10C14	B10C11	B10C4	B8A1	B8A2	B8A9			
C14	C11	C4	C14C14	C14C11	C14C4	C11D13	C11D12	C11D5			
D13	D12	D5	D13D13	D13D12	D13D5	D12F6	D12F7				
F6	F7					F7A1	F7A2	F7A9			

The blue-shaded cells are covered by the five existing state/transition cover tests

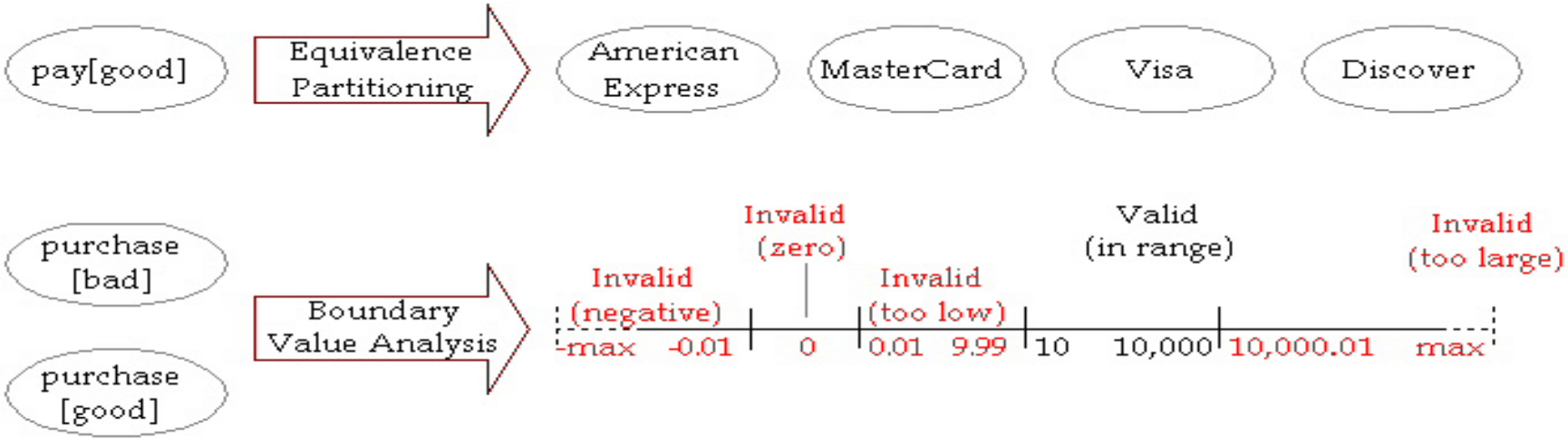
The red-shaded cells are covered by five new 1-switch cover tests

- (A1A1A2).
- (A9B8A1A9B8A2).
- (A9B10C14C14C4).
- (A9B10C11D13D13D5).
- (A9B10C11D12F7A1A9B10C11D12F7A9).

Copyright (c) RBCS 2008-2010

Equivalence Partitions and Boundary Values

- We can extend state-based testing with equivalence partitioning and boundary value analysis
- We combine transition, row, or transition sequence coverage with equivalence partition cover or boundary value coverage



www.rbcs-us.com
Copyright (c) RBCS 2008-2010

Decision Tables or State-based Testing?

- In some cases, a matter of taste
- Conditions can be used to model states if you want to use decision tables
- Test coverage is usually more thorough using state-based techniques
- In other cases, one technique or the other will clearly fit better
- Try both and see which feels most appropriate

Copyright (c) RBCS 2008-2010

Conclusion

- In this webinar, we've seen how to apply state transition diagrams, state transition tables, and switch coverage analysis to the testing of sophisticated and complex system
- These are especially useful for embedded systems and other systems where the system response to a given set of inputs depends on the past
- In our first webinar in this series, we looked at decision tables for testing detailed business rules
- We've now looked at state based methods to test state dependent systems
- In our next and final webinar in this series, we will look use cases
- This allows us to deal with the full range of internal business logic testing

Copyright (c) 2008-2010 RBCS

…Contact RBCS

For over a dozen years, RBCS has delivered services in consulting, outsourcing and training for software and hardware testing. Employing the industry's most experienced and recognized consultants, RBCS conducts product testing, builds and improves testing groups and hires testing staff for hundreds of clients worldwide. Ranging from Fortune 20 companies to start-ups, RBCS clients save time and money through improved product development, decreased tech support calls, improved corporate reputation and more. To learn more about RBCS, visit www.rbcs-us.com.

Address:	RBCS, Inc. 31520 Beck Road Bulverde, TX 78163-3911 USA
Phone:	+1 (830) 438-4830
Fax:	+1 (830) 438-4831
E-mail:	info@rbcs-us.com
Web:	www.rbcs-us.com

Copyright (c) RBCS 2008-2010-2010