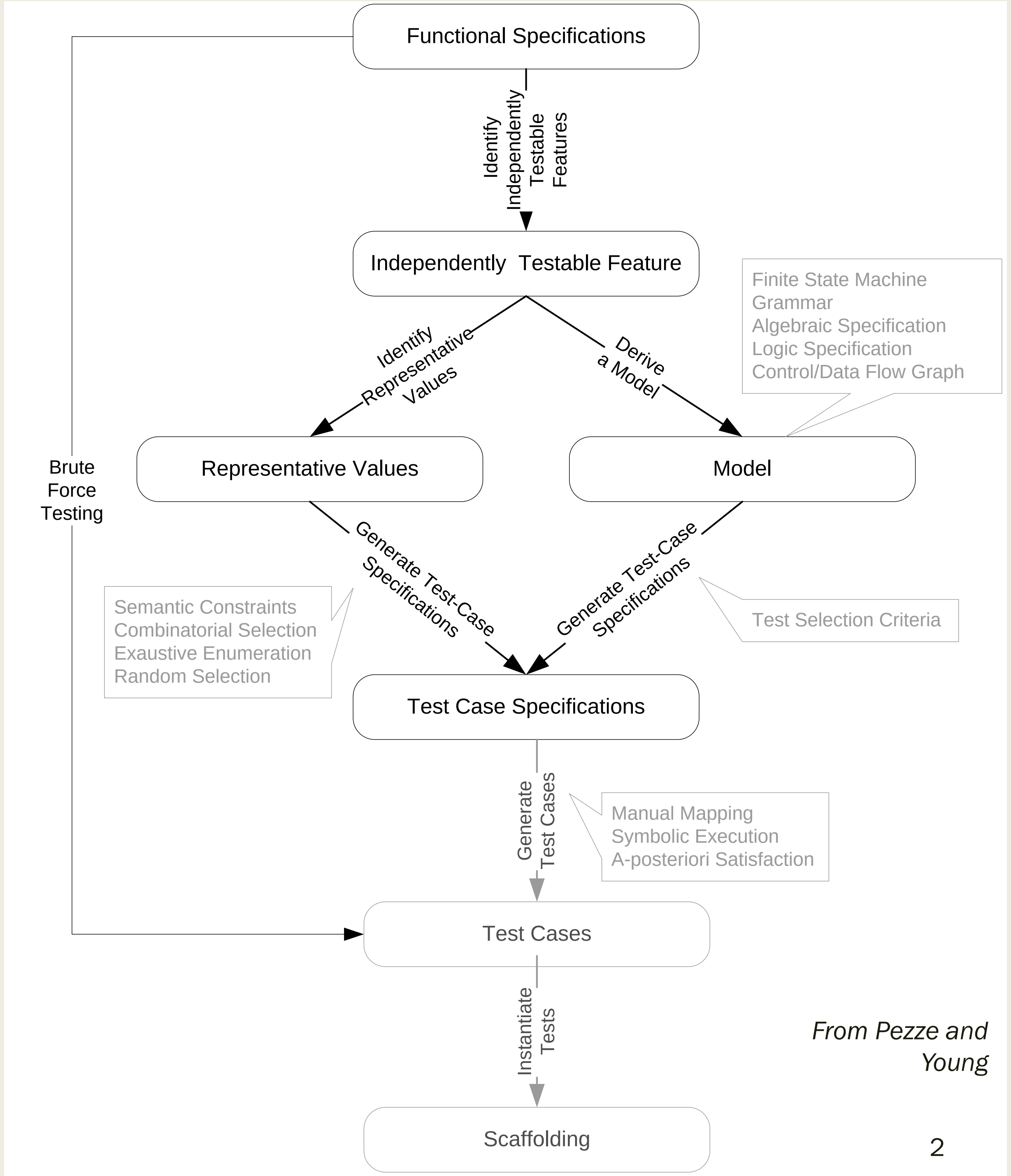
BLACK-BOX & WHITE-BOX

SENG 5811 – SPRING 2023 - Week #6

Tests from specifications

Going from requirements to tests



From Pezze and Young

Decision tables

- A compact way to model complex logic
- Each column corresponds to a Boolean combination of conditions
- Actions are the result/output when a combination is true
- Constraints may be specified that reduce the set of possible combinations

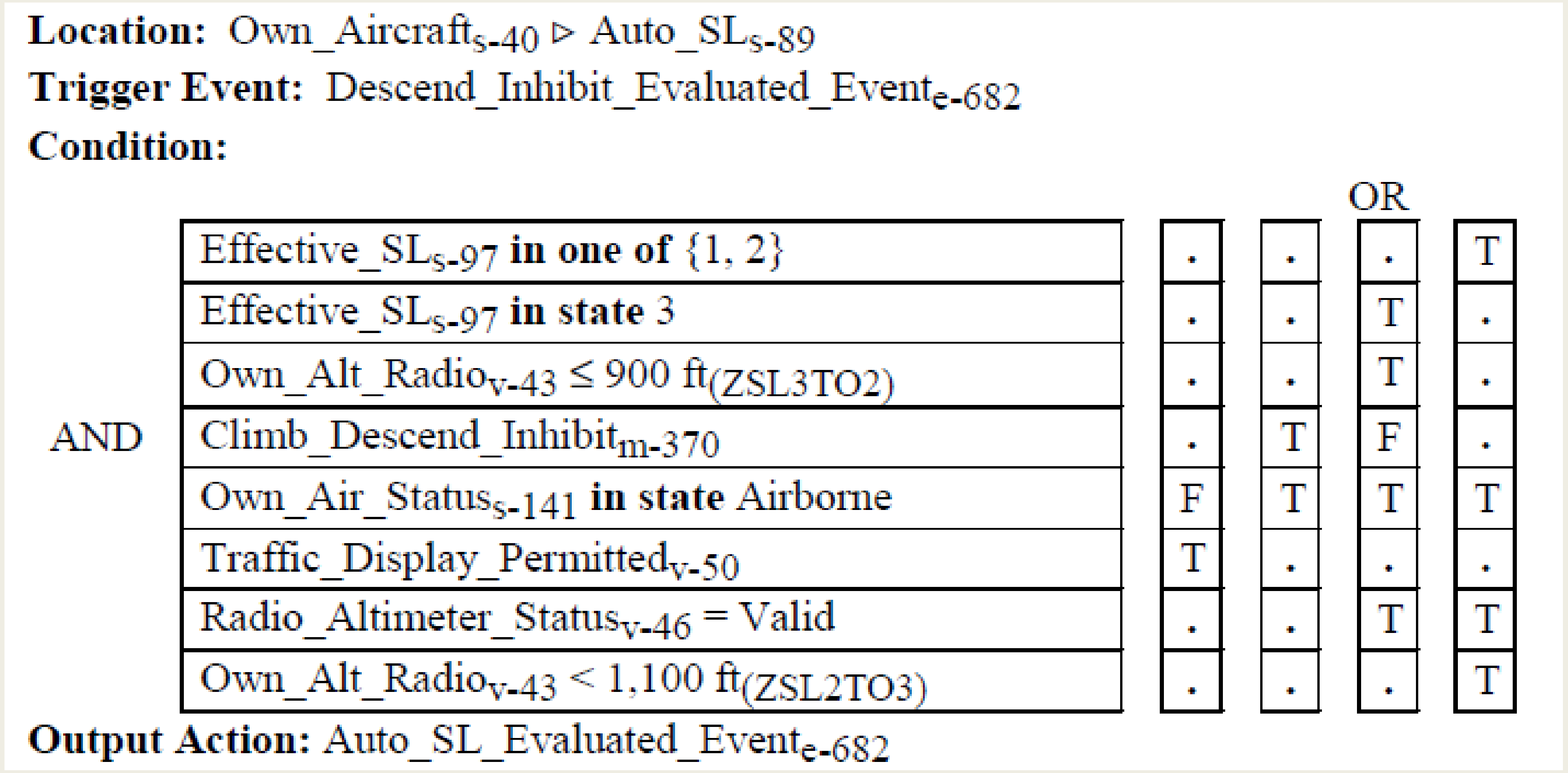
Conditions	…	Y	N	N	N
	…	-	N	Y	-
	…	-	Y	Y	N
	…	-	-	-	Y
Actions	…		X	X	X
	…	X		X	

Example: Decision table

	edu		individual					
EduAc	T	T	F	F	F	F	F	F
BusAc	-	-	F	F	F	F	F	F
CP > CT1	-	-	F	F	T	T	-	-
YP > YT1	-	-	-	-	-	-	-	-
CP > CT2	-	-	-	-	F	F	T	T
YP > YT2	-	-	-	-	-	-	-	-
SP < Sc	F	T	F	T	-	-	-	-
SP < T1	-	-	-	-	F	T	-	-
SP < T2	-	-	-	-	-	-	F	T
out	Edu	SP	ND	SP	T1	SP	T2	SP

at-most-one (EduAc, BusAc), at-most-one (YP < YT1, YP > YT2)
YP > YT2 -> YP > YT1, at-most-one (CP < CT1, CP > CT2),
CP > CT2 -> CP > CT1, at-most-one (SP < T1, SP > T2) SP > T2 -> SP > T1

Example: AND/OR Tables



Location: Own_Aircraft$_{s-40}$ ⊳ Auto_SL$_{s-89}$

Trigger Event: Descend_Inhibit_Evaluated_Event$_{e-682}$

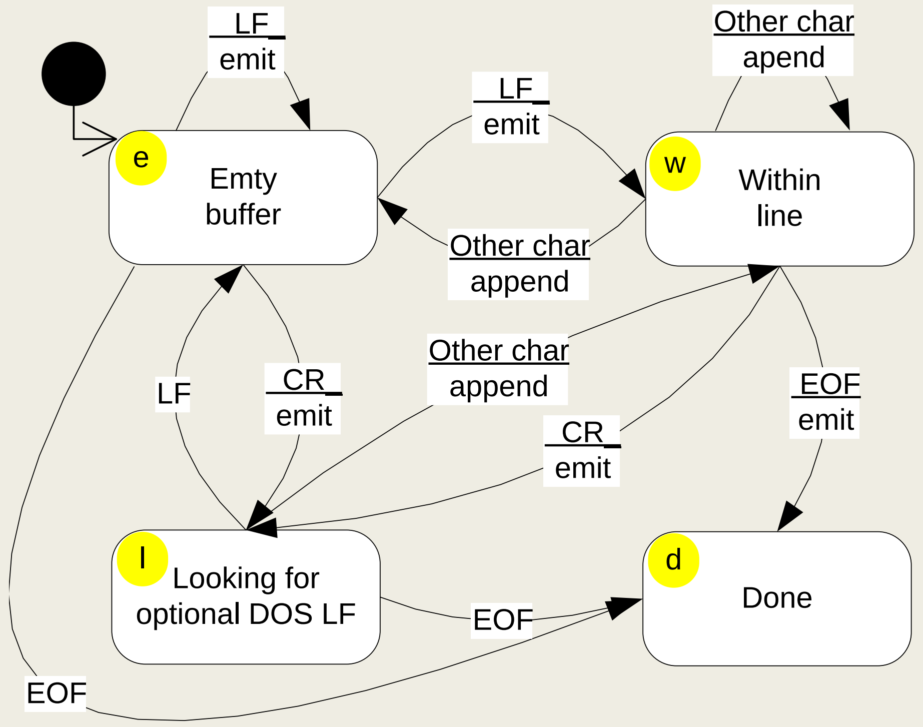
Condition:

				OR	
AND	Effective_SL$_{s-97}$ **in one of** {1, 2}	.	.	.	T
	Effective_SL$_{s-97}$ **in state** 3	.	.	T	.
	Own_Alt_Radio$_{v-43}$ $\leq$ 900 ft$_{(ZSL3TO2)}$	.	.	T	.
	Climb_Descend_Inhibit$_{m-370}$	.	T	F	.
	Own_Air_Status$_{s-141}$ **in state** Airborne	F	T	T	T
	Traffic_Display_Permitted$_{v-50}$	T	.	.	.
	Radio_Altimeter_Status$_{v-46}$ = Valid	.	.	T	T
	Own_Alt_Radio$_{v-43}$ < 1,100 ft$_{(ZSL2TO3)}$	.	.	.	T

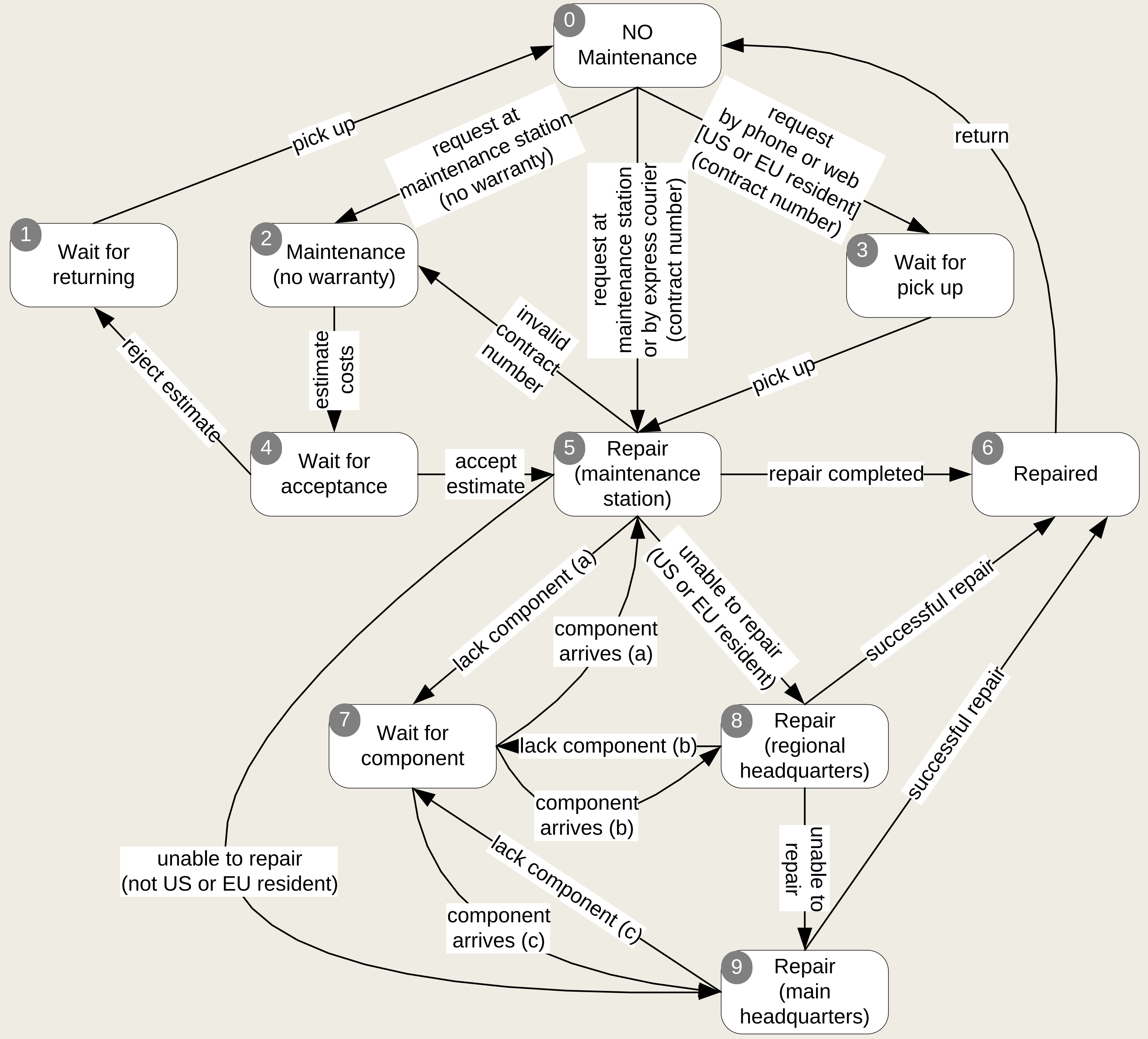
Output Action: Auto_SL_Evaluated_Event$_{e-682}$

Experiences From Specifying The TCAS II Requirements Using RSML.
Mats Heimdahl, Nancy Leveson, Jon Reese. Digital Avionics System Conference, Seattle, 1998.

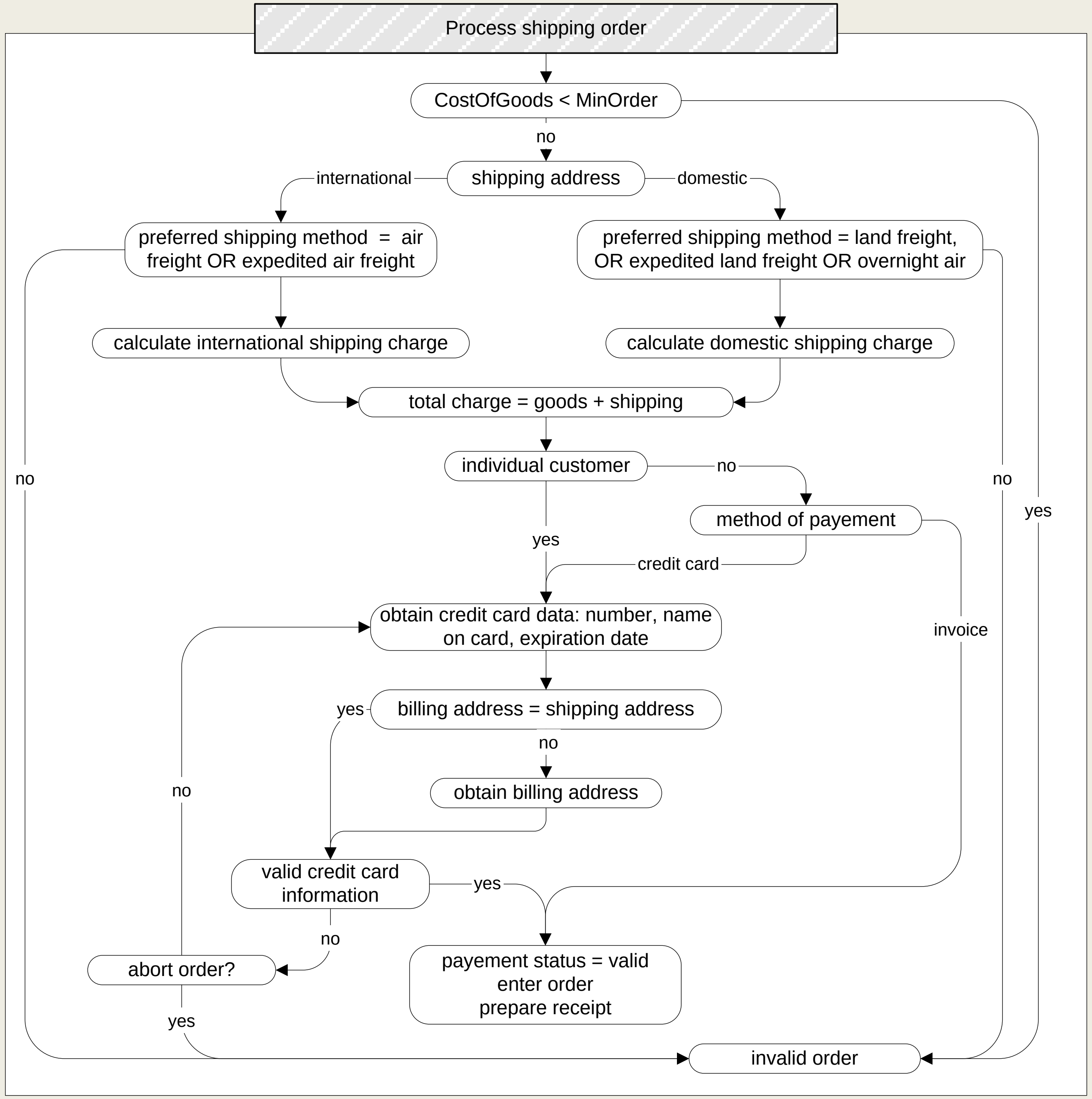
From State-Machines



	LF	CR	EOF	other
e	e/emit	e/emit	d/-	w/append
w	e/emit	e/emit	d/emit	w/append
l	e/-		d/-	w/append


0 NO Maintenance
1 Wait for returning
2 Maintenance (no warranty)
3 Wait for pick up
4 Wait for acceptance
5 Repair (maintenance station)
6 Repaired
7 Wait for component
8 Repair (regional headquarters)
9 Repair (main headquarters)
pick up
request at maintenance station (no warranty)
request at maintenance station or by express courier (contract number)
request by phone or web [US or EU resident] (contract number)
return
invalid contract number
estimate costs
reject estimate
pick up
accept estimate
repair completed
lack component (a)
component arrives (a)
unable to repair (US or EU resident)
successful repair
successful repair
lack component (b)
component arrives (b)
unable to repair
lack component (c)
component arrives (c)
unable to repair (not US or EU resident)

From Flow Graphs


Process shipping order
CostOfGoods < MinOrder
no
yes
shipping address
international
domestic
preferred shipping method = air freight OR expedited air freight
preferred shipping method = land freight, OR expedited land freight OR overnight air
no
no
calculate international shipping charge
calculate domestic shipping charge
total charge = goods + shipping
individual customer
no
method of payement
yes
credit card
invoice
obtain credit card data: number, name on card, expiration date
billing address = shipping address
yes
no
obtain billing address
valid credit card information
yes
no
abort order?
no
yes
payement status = valid enter order prepare receipt
invalid order

WHITE-BOX TESTING

White Box testing

- Tester can see the code
- Pseudonyms: Glass Box, Clear Box, Crystal Box, Structural Testing
- Provides much better code coverage than Black Box testing
- The following are visible to the tester:
  - *Boundaries*
  - *Control flow and Data flow*
  - *Complexity*

Examples of White Box Techniques

Dynamic Testing

- *Structural Coverage (statement, branch/ decision, condition)*
- *Control Flow analysis (path coverage)*
- *Data Flow analysis (data usage coverage)*
- *Code Profiling/Instrumentation*

Static Testing

- *Structured Evaluations*
- *Static Analysis*

Static Testing

- "Static": Not "Dynamic", i.e., you don't execute the code
- This includes some testing of documentation
- Two forms: Structured Evaluations [Section 4.1 in Spillner] and Static Analysis [Section 4.2 in Spillner]

Static Analysis

- Static Analysis uses Tools applies more to the software code than to documentation

Examples

- Compiling
- Compliance with Conventions and Standards
- Examination of the Data Flow graph
- Examination of the Control Flow graph
- Metrics (e.g. Cyclomatic Complexity)

Structural Coverage

- Coverage is a metric, not a method
- The other broad coverage category is Requirements Coverage
- Code that is implemented without being linked to requirements may not be exercised by requirements-based tests

Rationale for Structural Coverage

- Statistical approaches to quality assurance, which work well for physical devices, do not apply to software.
- Measure of the completeness of testing.

As an aside: using code coverage metrics to cross-check the Functional tests is a good idea but is generally hard to implement.

Purpose of Structural Coverage

- Unit Test use: Exercise the code; look for unintended functionality, incorrect behavior, unreachable code.

- Requirements Test use: Reveal code that has not been executed and which thus needs new test cases.

  Structural Coverage can improve the tests as much as it improves the code.

Covering Logic Expressions in the Software

- Logic expressions show up in many situations
- As just one example, covering logic expressions is required by the US Federal Aviation Administration for safety critical software
- Logical expressions occur in decisions or branch points in the software
- Tests are intended to choose some subset of the total number of truth assignments possible in the expressions

Types of Coverage

- Statement Coverage
- Branch (or Decision) Coverage
- Condition Coverage
- Multiple Condition Coverage
- Modified Condition/Decision Coverage (“MC/DC”)
- Path Coverage
  [Definitions of all of these are coming]

Points to Remember

Verifying these types of coverage is all well and good, but it serves no purpose unless you’re also verifying that, in each case, you get the program results that you want.

If your coverage tool says you got 100% coverage, but every single output was incorrect, then you have a problem!

Conditions and clauses

- A *condition* is an expression that evaluates to a Boolean value
- Conditions can contain
  - *Boolean variables*
  - *Relational operators: >, <, ==, >=, <=, !=*
  - *Boolean function calls*
  - *Boolean (logical) operators: AND, NOT, OR, etc.*
- A *clause* is a condition with no Boolean operators –this an "atomic (partial) condition"
  - *Each clause evaluates to either T or F.*
- A branch is a pathway out of a decision

Example

A Sample Condition…

$((a < b) \vee f(z)) \wedge D \wedge (m \geq n*o)$

… with four clauses:

1. $(a < b)$ – relational expression
2. $f(z)$ – Boolean-valued function
3. D – Boolean variable
4. $(m \geq n*o)$ – relational expression (with an arithmetical operator)

Logic Coverage Criteria Definitions

Statement or line coverage: Execute every line of code at least once [each node in the program graph]

Branch coverage: Execute each branch of every decision [Hence: “Decision coverage”]

Condition coverage: Execute every decision with each possible condition for each clause.

Multiple condition coverage: Execute each decision with all combinations of conditions for the clauses

Path coverage: Execute each path in the program graph

A Coverage Example

```
IF ( A < B AND C = 5) THEN
    DO {something};
SET D = 5;
```

Test cases you might think of:

a. *$A < B$, $C = 5$ [or constraint “(T, T)”]*
b. *$A < B$, $C \neq 5$ [or constraint “(T, F)”]*
c. *$A \geq B$, $C = 5$ [or constraint “(F, T)”]*
d. *$A \geq B$, $C \neq 5$ [or constraint “(F, F)”]*

A Coverage Example

IF (A < B AND C = 5) THEN

    DO {something};

SET D = 5;

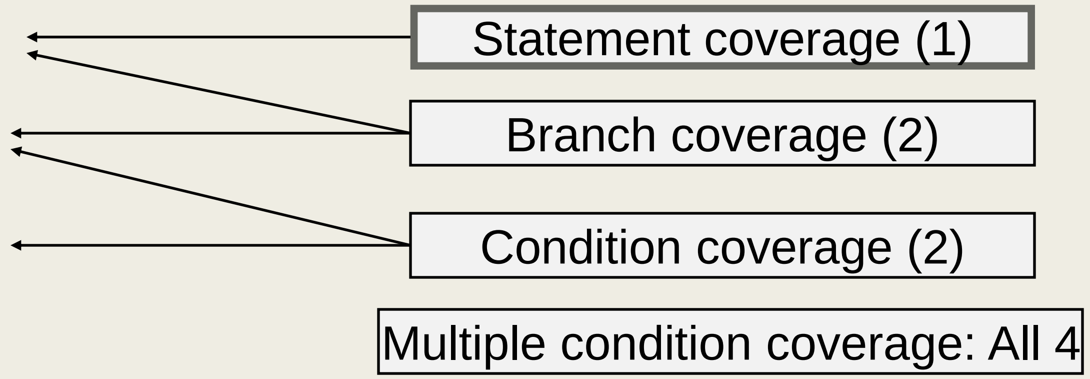
Associated test cases:

(a) $A < B$, $C = 5$

(b) $A < B$, $C \neq 5$

(c) $A \geq B$, $C = 5$

(d) $A \geq B$, $C \neq 5$



Example
[Inadequacy of Statement Coverage]

A C/C++ code fragment:

```
int* p = NULL;
if (some condition)
  p = &variable;
*p = 123;
```

Example
[Inadequacy of Branch Coverage]

```
if (A && (B || C))
   do-this;
else
   do-that;
```

A = FALSE executes do-that

A = B = TRUE executes do-this

We get Branch Coverage without even looking at C.

Example
[Inadequacy of Condition Coverage]

If Not (A or B) then C

Setting A = True, B = False, then A = False and B = True, satisfies Condition Coverage, but statement block C never gets executed;

i.e. we get neither statement coverage nor branch coverage!

[A has taken on the values T and F, and B has taken on the values T and F – that’s all]

The “Obvious” Solution: Multiple Condition Coverage

- Also called Combinatorial Coverage
- Requires that each possible combination of inputs be tested for each decision.
- Example: “if (A or B) then C” requires 4 test cases:

    A = True, B = True

    A = True, B = False

    A = False, B = True

    A = False, B = False

Multiple Condition Coverage

Multiple Condition Coverage does indeed give us Statement, Branch, and Condition Coverage.

The problem: For n conditions, 2^n test cases are needed, which grows exponentially with n

Controlling the Combinatorial Explosion

Modified Condition/Decision Coverage (MC/DC), or Condition Determination, of n conditions requires only n + 1 test cases.

A document on the topic:

"A Practical Tutorial on Modified Condition/Decision Coverage" http://shemesh.larc.nasa.gov/fm/papers/Hayhurst-2001-tm210876-MCDC.pdf

Modified Condition/Decision Coverage

- Requires that each condition be shown to independently affect the outcome of a decision, by varying one clause and leaving the other clauses fixed.
- Generally, for n conditions, MC/DC requires only n + 1 test cases
- Example: if (A or B) then C

3 test cases:	A = True, B = False (A rules)
	A = False, B = True (B rules)
	A = False, B = False (the *false* outcome)

Some MC/DC Examples

- To test If (A OR B):

A:	T	F	F
B:	F	T	F
Result:	T	T	F

- To test If (A AND B)

A:	F	T	T
B:	T	F	T
Result:	F	F	T

MC/DC Example

Which test cases below show each condition independently affecting the outcome? (How many should there be?)

Decision: X or Y or Z

X	Y	Z	Result
F	F	F	F
T	F	F	T
F	T	F	T
T	T	F	T
F	F	T	T
T	F	T	T
F	T	T	T
T	T	T	T

MC/DC Example

Which test cases below show each condition independently affecting the outcome?

Decision: X and Y and Z

X	Y	Z	Result
F	F	F	F
T	F	F	F
F	T	F	F
T	T	F	F
F	F	T	F
T	F	T	F
F	T	T	F
T	T	T	T

Next week

- Continue White-Box Testing – Data flow coverage
  - *No new reading: Review Chapter 7 of AO*

- Understand Mutation Analysis
  - *Read https://www.fuzzingbook.org/html/MutationAnalysis.html*
  - *If you are familiar with Python and Jupyter notebooks you can try out some of the code examples too, but it is not necessary*