Scalable Test Design using Ultra-Understandable Decision Tables

David Gelperin
Software Quality Engineering (www.sqe.com)
2425 Zealand Ave. N.
Golden Valley, MN 55427 U.S.A.
1-888-840-7563 +1-904-278-0707
sqegelp@aol.com

Abstract

This paper describes a scalable strategy that integrates multiple techniques into a pragmatic test design approach that honors budget and schedule constraints. The strategy is most appropriate for large sets of behavior rules and logical interpretations, where exhaustive testing is not feasible. Ultra-understandable decision tables and their supporting entity profiles [3] are used to describe behavior, but the design strategy can be applied to any equivalent information model. The scalable design process contains five steps: (1) Estimate affordable upper bound on size of test suite, (2) Select interpretations to be tested for each condition in a behavior rule, (3) Construct a rule selection strategy (4) Select foundation set of behavior rules to be tested, and (5) Apply selected interpretations to foundation set and remaining behavior rules. The operation of this strategy is demonstrated on life-like software.

Keywords

Specification-based testing, black-box testing, model-based test design, semi-formal models, ultra-understandable decision tables, entity profiles, dependency constraints, scalable test design, integrated test techniques, min/max true interpretations, min/max actions, all possible pairs, all effective values

1 Introduction

Although many individual test design techniques have been described in the literature [1, 2, 4, 5] and most descriptions council that the specified technique should be used in conjunction with others, there is little published advice on design strategies which both (1) integrate these techniques and (2) scale-down to provide test suites that respect budget and schedule constraints. Such strategies are needed for large functional testing problems i.e., those requiring several thousand tests for cost-effective coverage. In this paper, we describe such a strategy and demonstrate its operation on a life-like system. System behavior is modeled by semi-formal decision tables and supporting entity profiles, but can be applied to any equivalent information model.

Using decision tables rather than state tables for behavior specification implies behavior that is predominantly based on properties of the immediate input. For example, software containing input classification logic that makes little reference to stored data exhibits such behavior.

2 Prior Work

None of the individual design techniques described in this paper are new. Each has been explicitly or implicitly described in the literature: min true interpretations is described in [5], min/max actions is a form of boundary value testing [4], all pairs is described in [2], and effective values is implicit in Modified Condition/Decision Coverage [1].

What is new is (1) the focus on the pragmatics of integrating techniques and (2) the decision procedure for selecting various combinations of techniques in order to assure that the resulting test suite is as effective as possible within budget and schedule constraints.

3 Scalable Test Design

The scalable design process contains five steps:

1. Estimate affordable upper bound on size of test suite
2. Select interpretations to be tested for each condition in a behavior rule
3. Construct rule selection strategy
4. Select foundation set of behavior rules to be tested
5. Apply selected interpretations to foundation set and remaining behavior rules

We illustrate steps 2 through 5 with an incomplete, naïve example based on an actual operational system. The reader should quickly review the system described in the appendix and may wish to design a test suite using their current methods before reading the worked out example.

3.1 Estimate Upper Bound

The goal of suite bound estimation is to provide guidance to design decisions. This is not only to assure that the test suite respects the budget and schedule, but also to assure that candidate tests are not rejected because of phantom constraints.

Bounding the size of a test suite is an exercise in determining which order of magnitude is "too large". Is it ten, one hundred, one thousand, ten thousand, or one hundred thousand tests? An order of magnitude bound can be refined when enough is known about both the cost and time constraints and the cost and time requirements of each test. Costs include those of test development, execution, and maintenance. Time constraints include the time to develop and the time to execute.

3.2 Select Interpretations

An interpretation for the conditional expression of a behavior rule is the specific pattern of true and false conditions that causes that rule to be applicable to a situation. Interpretations can occur at multiple levels of abstraction. For example, consider the behavior rule:

Jane stays at home when the weather is very bad (C1)
    or she feels very sick (C2)
and the (condition definition for) weather is very bad
    when it is very cold (C1a) or snowing very hard (C1b)
    or raining very hard (C1c).

At the top level of abstraction, there are four possible interpretations for the rule's compound condition since C1 and C2 are independent. Of these interpretations, the compound condition is True for three of them (i.e., Jane stays at home) and False for the other as follows:

T1 – the weather is very bad and Jane feels very sick
T2 – the weather is very bad and Jane does not feel very sick
T3 – the weather is not very bad and Jane feels very sick
F1 – the weather is not very bad and Jane does not feel very sick

At the level of abstraction involving the definition of "the weather is very bad", there are eight interpretations, but only five are possible due to real world constraints. Of these five, the weather is very bad is True for four of them and False for the other as follows:

T1 – it is very cold and it is snowing very hard and (it is not raining very hard)
T2 – it is very cold and it is not snowing very hard and (it is not raining very hard)
T3 – it is not very cold and it is snowing very hard and (it is not raining very hard)
T4 – (it is not very cold) and (it is not snowing very hard) and it is raining very hard
F1 – it is not very cold and it is not snowing very hard and it is not raining very hard

Each of the conditions in parenthesis is True because of the truth of one of the other conditions in the interpretation.

Combining these two levels, there are ten possible interpretations for the extended conditional expression of the behavior rule, nine Trues and one False.

Min/Max True Interpretations Strategy

We now define a selection strategy for use when the number of true interpretations is too large. Consider behavior rule 3 in the Pass Orders System Decision Table.

Rule	Order is: Uncorrectable [3] Correctable [10] Valid [1]	Order is: Unoverrideable [1] OVerrideable [1] Authorized [1] OTher [1]	Order is: DOT [2] (Derivative Options Trading) Trader [2]	Order is: Unassigned [2] Assigned [1]		Receive order request & assign id & verify order	Support	Pass OK orders to DOT, Trader
3	Correctable	OVerrideable	Trader	Unassigned		X	correction overriding assignment	Trader

Designing a test involving this rule entails applying an interpretation in which Order is Correctable. To determine how many ways Order can be Correctable, we look at the derived profile of Customer Order and find:

Object Name	Compound Attribute Name	Attribute Description	Value Name	Value Definition
Customer Order	Correctness	Validity of all fields, but trader & authorizer		
			Correctable	At05 & At06 valid, but at least one of At02 through At04 and At07 through At12 not valid

Therefore Order is Correctable when
[Customer Identifier is valid]
and [Security Identifier is valid]
and [(Order Date is invalid) or (Order Time is invalid) or (Order Taker is invalid)
or (Order Type is invalid) or (Order Quantity is invalid)
or (Order Duration is invalid) or (Security Price Type is invalid)
or (Security Price Limit is invalid) or (Trade Payment Type is invalid)] .

If we assume that each condition in this definition has exactly one way to be True and one way to be False, there are 2^{11} (= 2048) interpretations of which $2^9 - 1$ (= 511) are True and therefore 511 distinct ways in which Order can be Correctable.

The **minimum true interpretations strategy** selects just those interpretations in which exactly one of the disjuncts in the disjunctive normal form of the conditional expression is True. For Order is Correctable, there are 9 such interpretations.

The **min/max true interpretations strategy** includes these interpretations plus one in which a maximum number of disjuncts in the disjunctive normal form of the conditional expression is True. This strategy applied to Order is Correctable results in 10 interpretations of which two are:

[Customer Identifier is valid]
and [Security Identifier is valid]
and [(Order Date is invalid)
and (Order Time is valid)
and (Order Taker is valid)
and (Order Type is valid)
and (Order Quantity is valid)
and (Order Duration is valid)
and (Security Price Type is valid)
and (Security Price Limit is valid)
and (Trade Payment Type is valid)]

[Customer Identifier is valid]
and [Security Identifier is valid]
and [(Order Date is invalid)
and (Order Time is invalid)
and (Order Taker is invalid)
and (Order Type is invalid)
and (Order Quantity is invalid)
and (Order Duration is invalid)
and (Security Price Type is invalid)
and (Security Price Limit is invalid)
and (Trade Payment Type is invalid)]

These are the results if each condition has only one way to be True and one to be False. But, what if some conditions have more than two options. For example, looking at the definition of Order Quantity in the basic profile, we find:

Object Name	Attribute Name	Attribute Description	Attribute Identifier	Value Name	Value Definition
Customer Order	Order Identifier	Unique identifier	At01		
	Order Quantity	Number of security units to be traded	At08	Valid	0 < value

Order Quantity is a numeric attribute whose value must be greater than 0. Therefore, this attribute can be invalid by being non-numeric or by being equal to 0. If just this attribute has two distinct invalid outcomes, then the total number of interpretations increases by 1024 to 3072, the number of True interpretations increases by 256 to 767, and the number of min/max interpretations increases by 1 to 11.

The min/max true interpretations strategy would be used for each simple and compound attribute value associated with the set of behavior rules. For the behavior rules of the Pass Orders System, there are 11 such attribute values at the top of the decision table.

3.3 Construct a Rule Selection Strategy

To support scaling, we describe two types of selection criteria with options for each. The first set of options entails output-oriented criteria and the second adds input-oriented criteria. Each type of criteria includes the no choice option. A rule selection strategy would be constructed by choosing one option from each type.

3.3.1 Output-Oriented Criteria

In a set of behavior rules, each action is either conditional (i.e., sometimes does not occur) or unconditional (i.e., always occurs). Output-oriented criteria focus on conditional actions and their co-occurrences.

3.3.1.1 Output Option 1: Min/Max Actions + All Possible Action Pairs

Following the approach to min/max interpretations, we define a **min/max actions criterion** in an analogous fashion. For each conditional action, the minimum actions criterion selects one rule which has the smallest number of co-occurring conditional actions. This selection may not be unique. For each conditional action, the maximum actions criterion selects one rule which has the greatest number of co-occurring conditional actions. Again, the selection may not be unique. The min/max actions criteria combines these two approaches.

For the Pass Orders System, rules 1-2, 8-9, and 12-14 satisfy the minimum actions criterion, while rules 1-2, and 3-4 satisfy the maximum actions criterion. Therefore, the set of 9 rules 1-4, 8-9, and 12-14 satisfies the min/max actions criteria.

Some conditional actions can co-occur, while others can not (e.g., support assignment and pass to DOT). The **all-possible action pairs criterion** requires that all conditional actions that can co-occur must be in the selected set

For the Pass Orders System, rules 3-4 satisfy this criterion. For other rule sets, this criterion would add rules to the min/max actions set.

3.3.1.2 Output Option 2: Maximum Actions or All Possible Action Pairs

If the set from option 1 is too large, one or both of these criteria could yield an acceptable size set.

3.3.1.3 Output Option 3: All Risky Action Pairs

This criterion assumes that knowledge about failure-impact or fault-likelihood risk is available to guide the selection process and that most action pairs are not risky. This criterion yields a smaller suite that is cost-effective as long as the risk knowledge is accurate.

3.3.2 Input-Oriented Criteria

To define these criteria, we introduce the concept of **affective input values or conditions** relative to a behavior rule. Consider the following behavior rules:

Rule	Order is: Uncorrectable [3] Correctable [10] Valid [1]	Order is: Unoverrideable [1] OVerrideable [1] Authorized [1] OTher [1]	Order is: DOT [2] (Derivative Options Trading) Trader [2]	Order is: Unassigned [2] Assigned [1]	Receive order request & assign id & verify order	Support	Pass OK orders to DOT, Trader	Pass NG orders to Head Trader
1	Uncorrectable	---	---	---	X			Uncorrect-able
10	Correctable	Authorized or OTher	Trader	Assigned	X	correction	Trader	
14	Valid	Authorized or OTher	Trader	Assigned	X		Trader	

Note that Order is Correctable in rule 10 and that changing this condition to either of its alternatives will change the execution result in any correct implementation of these rules. This is the test for value or condition affectiveness. If changing a value or condition to any of its alternatives, while holding all other conditional elements of the rule constant, causes a change in the execution result, then the original value or condition is affective relative to this rule. Note that Order is Trader in rule 1 is not affective by this test.

Now consider Order is Authorized in rule 10. This is also not affective by this definition, since changing it to Order is OTher yields the same execution result in any correct implementation. However changing it to either Order is Unoverrideable or Order is OVerrideable yields a different result. To describe this situation, we speak of **an affective set of input values or conditions**. In our example, Authorized and OTher constitute an affective set. For affective sets, changing to any alternative outside the set changes the result, while changing to any alternative in the set does not.

The definitions for affective values, conditions, and sets hold for inputs affecting calculations as well as those affecting selection.

3.3.2.1 Input Option 1: All Possible Affective Pairs + All Effective Values

Two values or conditions appearing in a behavior rule are an affective pair if and only if they are each affective or a member of an effect set in that rule.

The **all-possible affective pairs** criterion requires that every pair which is effective somewhere in a set of behavior rules must appear as an affective pair in the selected subset. For the Pass Orders System, rules 2-4 and 10-13 satisfy this criterion.

The **all effective values criterion** requires that each value or condition which is effective or a member of an effect set somewhere in a set of behavior rules must appear and be effective in the selected subset. For the Pass Orders System, rules 1-3, 13, and 14 satisfy this criterion.

The combined criteria are satisfied by rules 1-4 and 10-13. Note that including the effective values criterion adds rules in which there is only one effective value and that value is not contained in an affective pair in any rule.

3.3.2.2 Input Option 2: All Effective Values

If the set from option 1 is too large, using just the effective values criterion could yield an acceptable size set.

3.3.3 Using the Criteria

If the total number of rules is well within the upper bound, then all rules should be chosen. If there are "too many" rules, then the choices must be scaled down to satisfy budget and schedule constraints by choosing one option from each type of selection criteria.

Choosing option 1 from both the input and output criteria yields the largest foundation set produced by this method when there are too many rules. For purposes of illustration, we make this choice although the rule set is clearly small enough to choose all rules.

3.4 Select Foundation Set

The next step is to choose the set of behavior rules to be used as the foundation of the test suite. While test cases may be based on rules outside the foundation set – as we will see in the next section, every rule in the foundation set will be the basis for some test.

In our example, we could just merge the rule sets from each option 1 to yield 1-4 and 8-14. However, since some of those rule selections were not unique, we find that 1-4 and 10-14 are sufficient to satisfy both criteria. This then is our foundation set.

Rule	Order is: Uncorrectable [3] Correctable [10] Valid [1]	Order is: Unoverrideable [1] OVerrideable [1] Authorized [1] OTher [1]	Order is: DOT [2] (Derivative Options Trading) Trader [2]	Order is: Unassigned [2] Assigned [1]	Receive order request & assign id & verify order	Support	Pass OK orders to DOT, Trader	Pass NG orders to Head Trader
1	Uncorrectable	---	---	---	X			Uncorrect-able
2	Valid or Correctable	Unoverrideable	---	---	X			Unoverride-able
3	Correctable	OVerrideable	Trader	Unassigned	X	correction overriding assignment	Trader	
4	Correctable	OVerrideable	DOT	(Trader Id is empty i.e., unassigned)	X	correction overriding	DOT	
10	Correctable	Authorized or OTher	Trader	Assigned	X	correction	Trader	
11	Valid	OVerrideable	Trader	Assigned	X	overriding	Trader	
12	Valid	Authorized or OTher	Trader	Unassigned	X	assignment	Trader	
13	Valid	Authorized or OTher	DOT	(Trader Id is empty i.e., unassigned)	X		DOT	
14	Valid	Authorized or OTher	Trader	Assigned	X		Trader	

3.5 Apply Interpretations

The last step in our design process is to apply the selected logical interpretations to the foundation set of rules and perhaps others as well. We will not describe the entire process for our example, but will illustrate the essential elements of this design step.

We begin with the condition having the greatest number of interpretations. In our example, this is Order is

Correctable. Let us assume that we have decided that testing 10 distinct interpretations for this condition is sufficient. Our problem is that there are only 4 opportunities to do this in the foundation set. We could just replicate one or more of these 4 rules until we had 10 sites, but since we need 6 more tests anyway, we choose new rules from outside the set in order to test more behavior patterns. It turns out that rules 5, 6, and 8 also provide opportunities, so we include them. We still need to replicate 3 of these 7 rules to provide the remaining sites. The choices made during the process of selecting outside the foundation set or choosing which rule to replicate may be influenced by the upcoming need to site interpretations for other conditions – especially those with large numbers.

We reapply this process to the interpretation with the next greatest number until finally all selected interpretations have been sited. This ends the process and provides our suite of tests. An example of a final result is:

Test	Order is: Uncorrectable [3] Correctable [10] Valid [1]	Order is: Unoverrideable [1] OVerrideable [1] Authorized [1] OTher [1]	Order is: DOT [2] (Derivative Options Trading) Trader [2]	Order is: Unassigned [2] Assigned [1]		Receive order request & assign id & verify order	Support	Pass OK orders to DOT, Trader	Pass NG orders to Head Trader
1	Uncorrectable 1	---	---	---		X			Uncorrect-able
2	Uncorrectable 2	---	---	---		X			Uncorrect-able
3	Uncorrectable 3	---	---	---		X			Uncorrect-able
4	Correctable 1	Unoverrideable	---	---		X			Unoverride-able
5	Correctable 2	OVerrideable	Trader 1	Unassigned 1		X	correction overriding assignment	Trader	
6	Correctable 3	OVerrideable	DOT 1	(Trader Id is empty i.e., unassigned)		X	correction overriding	DOT	
7	Correctable 4	OVerrideable	Trader 2	Assigned		X	correction overriding	Trader	
8	Correctable 5	Authorized	Trader ?	Unassigned 2		X	correction assignment	Trader	
9	Correctable 6	OTher	DOT 2	(Trader Id is empty i.e., unassigned)		X	correction	DOT	
10	Correctable 7	OTher	Trader ?	Assigned		X	correction	Trader	
11	Correctable 8	Authorized	Trader ?	Assigned		X	correction	Trader	
12	Correctable 9	OTher	Trader ?	Assigned		X	correction	Trader	
13	Correctable 10	Authorized	Trader ?	Assigned		X	correction	Trader	
14	Valid	OVerrideable	Trader ?	Assigned		X	overriding	Trader	
15	Valid	OTher	Trader ?	Unassigned ?		X	assignment	Trader	
16	Valid	Authorized	DOT ?	(Trader Id is empty i.e., unassigned)		X		DOT	
17	Valid	Authorized	Trader ?	Assigned		X		Trader	

If the number of tests grows too large, we must reevaluate our decisions about selecting interpretations, constructing the rule selection strategy, or both.

References

[1] John Joseph Chilenski and Steven P. Miller. Applicability of Modified Condition/Decision Coverage to Software Testing. Software Engineering Journal, 9(5):193-200, September 1994.

[2] David Cohen, Siddhartha Dalal, Jesse Parelius, and Gardner Patton. The Combinatorial Approach to Automatic Test Generation. IEEE Software 13(5): 83-88 September 1996

[3] David Gelperin. Ultra-Understandable Logic Tables. submitted for publication and available at www.stqe.net in the interest areas - software testing – tools/automation – library

[4] Glenford Myers. The Art of Software Testing. Wiley 1979

[5] E. Weyuker, T. Goradia, and A. Singh. Automatically generating test data from a Boolean specification. IEEE Transactions on Software Engineering, 20(5):353-363, May 1994

Appendix -- Elements of the Pass Orders System Logic Model

1. Abstract of Pass Orders System

This system is a partially automated front-end to a set of financial trading systems. The system handles orders for stock and bonds as well as options (e.g., puts and calls).

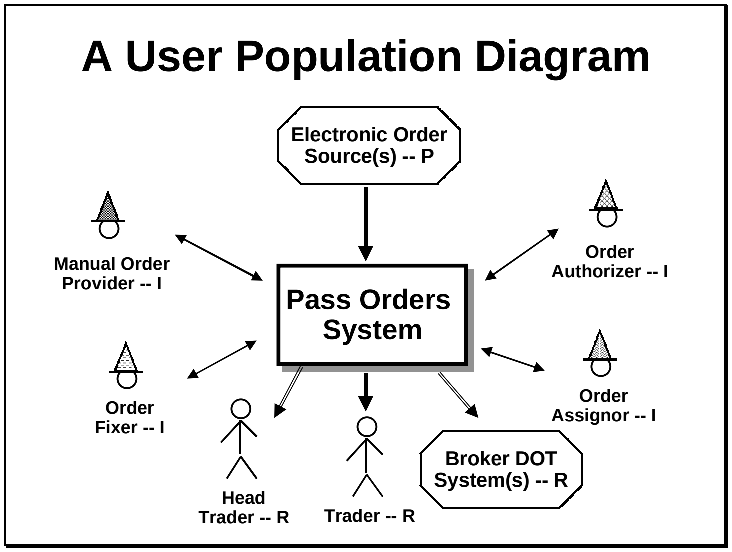
The system is fed by electronic and manual order sources and sends output to Traders, Head Traders, and the Derivative Options Trading (DOT) system.

It interacts with humans who provide manual orders as well as fix, authorize, or assign the electronic or manual orders.

The system:

1. accepts either new manual, new electronic, or modified orders,
2. assigns an order number to new orders,
3. verifies each order,
4. supports the fixing of invalid or unauthorized orders that can be fixed,
5. supports the assignment to a Trader of proper, but unassigned, orders for stocks or bonds,
6. rejects uncorrectable or unoverrideable orders to a Head Trader,
7. passes proper options orders to the DOT system, and
8. passes proper, assigned orders for stocks or bonds to a Trader

2. User Population Diagram in Pass Orders System Logic Model



3a. Basic Profile of Customer Order in Pass Orders System Logic Model

Object Name	Attribute Name	Attribute Description	Attribute Identifier	Value Name	Value Definition
Customer Order	Order Identifier	Unique identifier	At01		
	Order Date	Date order taken	At02		
	Order Time	Time order taken	At03	Valid	$0 \leq$ value ≤ 2359
	Order Taker	Id of order taker	At04		
	Customer Identifier	Unique identifier	At05		
	Security Identifier	Unique identifier	At06		
	Order Type	Code for type of order	At07	Buy Buy to Cover Sell	Buy BuyC Sell
	Order Quantity	# of units to be traded	At08	Valid	0 < value
	Order Duration	Code for time order is valid	At09	Day 'til Canceled Fill or Kill On the Open	Day TCan ForK Open
	Security Price Type	Code for type of price	At10	Market Limit Stop Loss	Mark Limt Stop
	Security Price Limit	Bound on trade price	At11		
	Trade Payment Type	Code for type of payment	At12	Cash Margin	C M
	Trader Identifier	Unique identifier	At13	Assigned	Not Empty
	Authorizer Identifier	Unique identifier	At14		

3b. Derived Profile of Customer Order in Pass Orders System Logic Model

Object Name	Derived Attribute Name	Attribute Description	Value Name	Value Definition
Customer Order	Correctness	Validity of all fields, but trader & authorizer		
			Valid	At02 through At12 all valid
			Correctable	At05 & At06 valid, but at least one of At02 through At04 and At07 through At12 not valid
			Uncorrectable	At05 or At06 not valid
	Authorization	Order authorization situation		
			Authorized	At14 is (non-empty & valid)
			Overrideable	Order is valid and At14 is invalid, but order value < 50K
			Unoverrideable	At14 is invalid, but order value ≥ 50K
			Other	At14 is (empty & valid)
	Assignment	Assignment to trader situation		
			Assigned	At13 is (non-empty & valid)
			Unassigned	At13 is (non-empty & invalid) or empty

3c. Basic Profile of Security in Pass Orders System Logic Model

Object Name	Attribute Name	Attribute Description	Attribute Identifier	Value Name	Value Definition
Security	Security Identifier	Unique identifier	At01		
	Security Type	Code for type of security	At02	Bond / Stock Put / Call	BOND / STCK PUT / CALL
	Exchange	Place to trade	At03	NYSE Amex NASDAQ Non-US	NYSE AMEX NASD NOUS
	Recent Price	Price of a unit	At04		

3d. Derived Profile of Security in Pass Orders System Logic Model

Object Name	Derived Attribute Name	Attribute Description	Value Name	Value Definition
Security	Derivative?	Principal or Derivative Type of Security		
			Derivative	Security Type is Put or Call
			Principal	Security Type is Stock or Bond

3e. Condition Dependency Constraints in Pass Orders System Logic Model

Implying Conditions	Dependency Type	Implied Conditions
Security is Derivative	←→	Not (Trader is Assigned)

4a. Basic Reaction Dictionary in Pass Orders System Logic Model

Generic Name	Specific	Pre-Conditions	Post-Conditions
Receive order request			
	Electronic	Electronic order submitted	Electronic order received & Electronic count up 1
	Manual	Manual order submitted	Manual order received & Manual count up 1
	Modified	Modified order submitted	Modified order received
Assign id		Electronic or Manual Order received	Order id assigned
Verify order		Order id assigned or Modified order received	[Order is valid EOR Order is correctable EOR Order is uncorrectable] & [Order is authorized EOR Order is other EOR Order is overrideable EOR Order is unoverrideable] & [Order is assigned EOR Order is unassigned] & [Order is trader EOR Order is DOT]
Display problem order			
	Invalid	Order is correctable	Order is displayed
	Unauthorized	Order is overrideable	Order is displayed
	Unassigned	Order is trader & unassigned	Order is displayed
Pass order			
	Trader	Order is valid, (authorized or other), assigned, and trader	Trader count up 1
	DOT system	Order is valid, (authorized or other), and DOT	DOT count up 1
	Head Trader	Order is uncorrectable or unoverrideable	Rejected count up 1

4b. Derived Reaction Dictionary in Pass Orders System Logic Model

Derived Reaction	Basic Reactions
Support correction	Display invalid order Receive modified order request Verify modified order
Support overriding	Display unauthorized order Receive modified order request Verify modified order
Support assignment	Display unassigned order Receive modified order request Verify modified order

5. Ultra-Understandable Decision Table in Pass Orders System Logic Model

Rule	Order is: Uncorrectable [3] Correctable [10] Valid [1]	Order is: Unoverrideable [1] OVerrideable [1] Authorized [1] OTher [1]	Order is: DOT [2] (Derivative Options Trading) Trader [2]	Order is: Unassigned [2] Assigned [1]	Receive order request & assign id & verify order	Support	Pass OK orders to DOT, Trader	Pass NG orders to Head Trader
1	Uncorrectable	---	---	---	X			Uncorrect-able
2	Valid or Correctable	Unoverrideable	---	---	X			Unoverride-able
3	Correctable	OVerrideable	Trader	Unassigned	X	correction overriding assignment	Trader	
4	Correctable	OVerrideable	DOT	(Trader Id is empty i.e., unassigned)	X	correction overriding	DOT	
5	Correctable	OVerrideable	Trader	Assigned	X	correction overriding	Trader	
6	Correctable	Authorized or OTher	Trader	Unassigned	X	correction assignment	Trader	
7	Valid	OVerrideable	Trader	Unassigned	X	overriding assignment	Trader	
8	Correctable	Authorized or OTher	DOT	(Trader Id is empty i.e., unassigned)	X	correction	DOT	
9	Valid	OVerrideable	DOT	(Trader Id is empty i.e., unassigned)	X	overriding	DOT	
10	Correctable	Authorized or OTher	Trader	Assigned	X	correction	Trader	
11	Valid	OVerrideable	Trader	Assigned	X	overriding	Trader	
12	Valid	Authorized or OTher	Trader	Unassigned	X	assignment	Trader	
13	Valid	Authorized or OTher	DOT	(Trader Id is empty i.e., unassigned)	X		DOT	
14	Valid	Authorized or OTher	Trader	Assigned	X		Trader	