How to Perform System-to-System Integration Testing with Test Case Designer

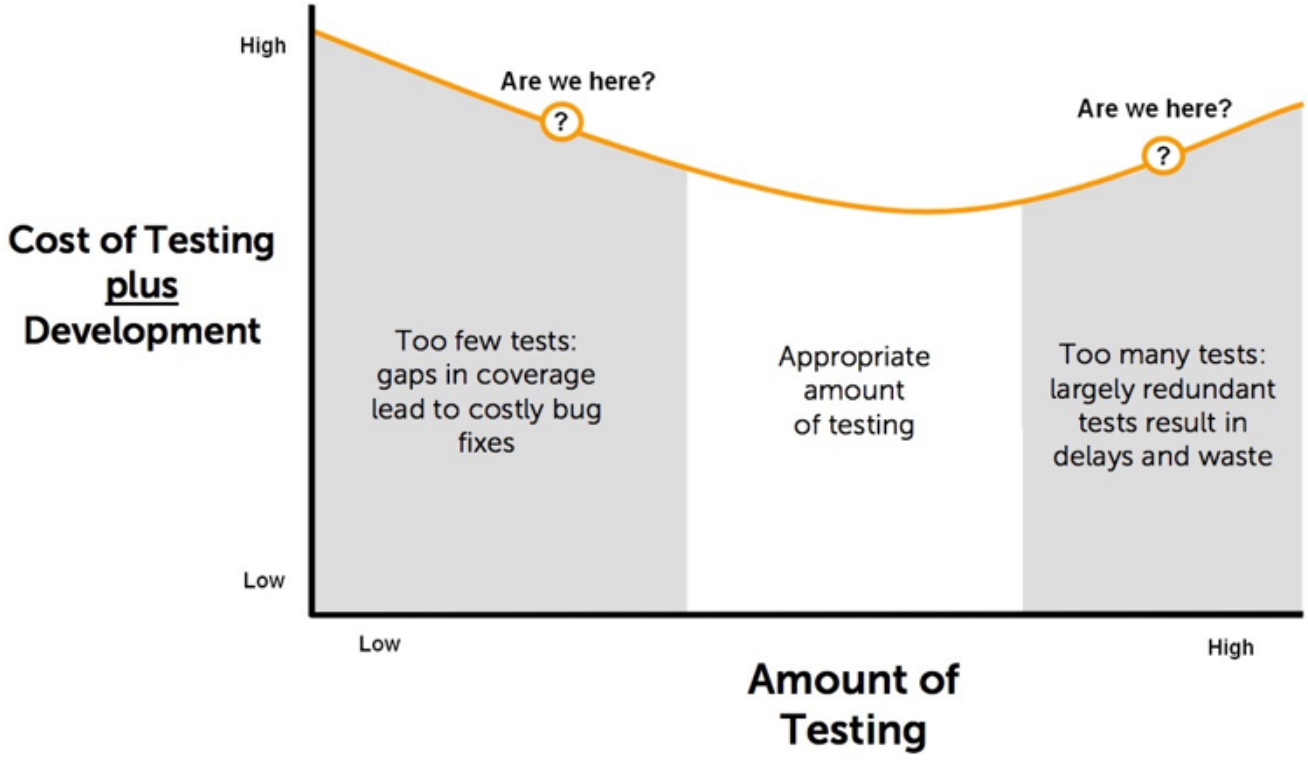
Learn about optimizing coverage, traceability, and the integration scenario count in TCD, taking wealth management as an example.

- Core Testing Challenge
- Test Case Designer Solution & Modeling Process
  - Building a Model – Step 1 – Parameter & Value Selection
  - Building a Model – Step 2 – Generating Optimal Scenarios
  - Building a Model – Step 3 – Coverage Results Comparison
  - Building a Model – Step 4 – Scripting & Export
- Summary & Case Studies

Core Testing Challenge

While functional testing is a common entry point for Test Case Designer implementations, and we have multiple articles describing the benefits achieved, integration testing (both system-to-system and E2E) is often even more applicable because of the increase in scale and several dependencies (and consequently, in several important possible interactions).

However, with that increase comes greater difficulty in decision making which leads to the major challenge in testing complex systems – **"how much testing is enough?"**



In such situations, teams use TCD to quickly determine the optimal answers to both (1) "how many tests?" and (2) "which specific tests?".

We will describe how these benefits come to life on the generalized example of a wealth management implementation where System A handles risk profile evaluation (like this), System B – opening an account (like this), and System C – places the trade.

Test Case Designer Solution & Modeling Process

Building a Model – Step 1 – Parameter & Value Selection

The general logic of “Parameter = a step in the flow that impacts system outcomes and can be represented with finite variation” still applies throughout the article, the notes below build on top of it.

The general rules for system-to-system parameter & value identification:

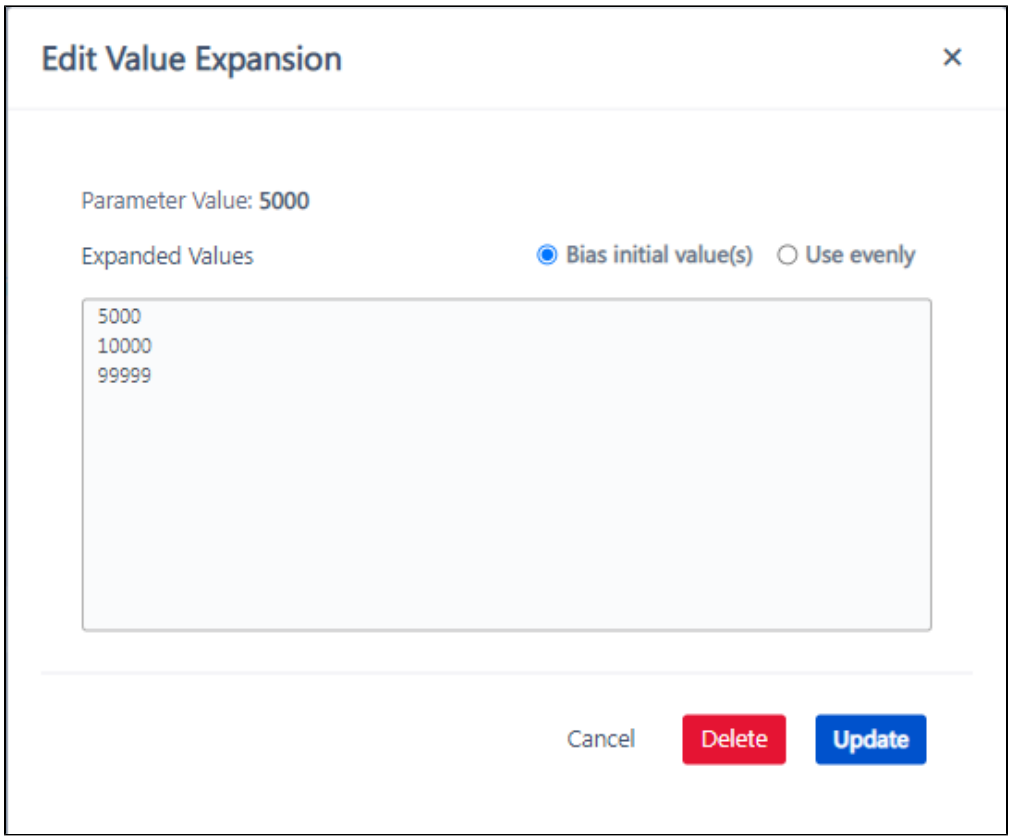
1. prioritize parameters that affect more than 1 system;
2. prioritize values heavily involved in business rule/integration triggers;
3. prioritize value expansions for less impactful options.

In the financial implementations, each system is usually contributing a “fair share” of parameters to the TCD model. The key characteristic is the model shape – more “horizontal,” with fewer factors but larger value lists (Claims Processing in Insurance is another area with similar model shapes, unlike the typically “long and narrow” policy side).

System A (1)	******					
Where do you see yourself with regards t...	1 - Conservative	2	3	4	5 - Aggressive	
One-time opportunity question 1 (2)	A	B				
One-time opportunity question 2 (2)	A	B				
Experience of financial losses (4)	not emotionally affected	uncomfortable but not at risk	uncomfortable and afraid	none		
Substantial loss reaction (4)	immediately switch	wait three months	wait one year	stay with the current investme...		
System A v2 outcome - User Profile (6)	1	2	3	4	5	6
System B (1)	******					
Account Type (3)	Self-service Brokerage Account	Dedicated Financial Advisor	Mix			
Order Type (8)	Market	Limit	Stop / Limit	CAB	Debit	Credit
Initial Investment (4)	0	5000	100000	> 100000		
Fees (3)	0	0.35	0.4 - 0.6			
Communication Preferences (2)	Paper	Electronic				
System B outcome 1 (1)	User is able to open the account					

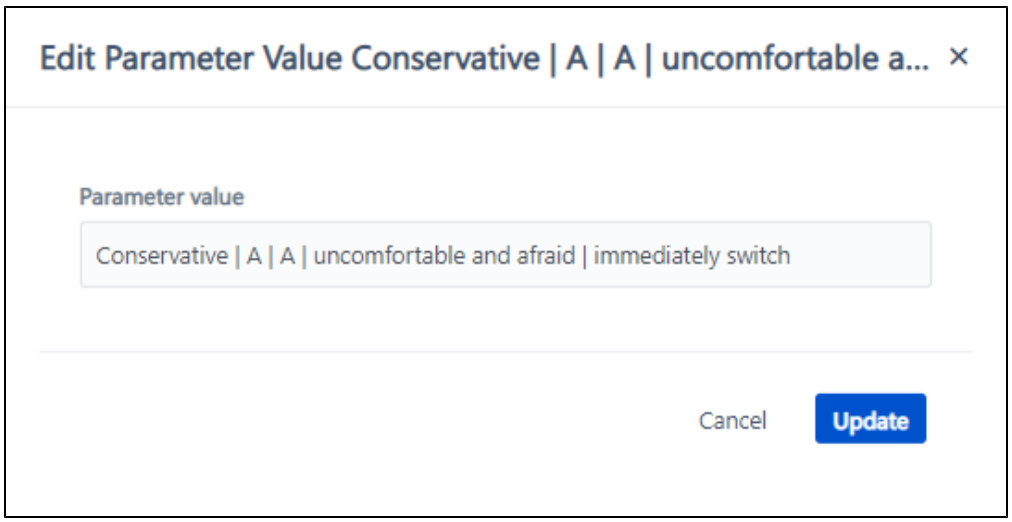
System-to-system modeling is not likely to hit the 256 parameter limit in TCD, so you have the flexibility of including less important factors, but are necessary for “passive regression” traceability and/or automation (and could increase your defect prevention).

To make sure you both prioritize boundary values and explore options beyond them, you can use the “bias” feature (left radio button) within the value expansion dialog:



The interactions between all the sub-elements of each system are often less of a focus, so sections of the model could have the “major outcome” parameters (e.g., User Profile, Account Type) which will help with the algorithm selections on the next modeling step described later.

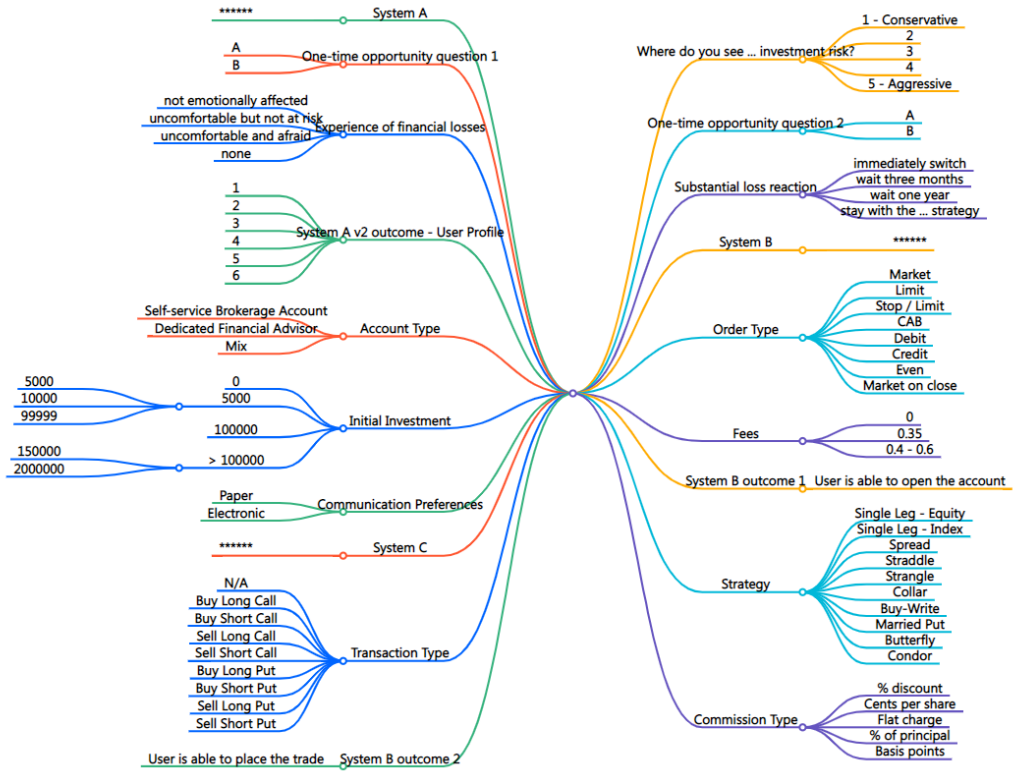
If for any reason including factors other than the “major outcome” is not feasible/relevant, you can leverage the “nested” values where the name would describe all the contextual characteristics (instead of just saying “1”):



That way, any reviewer could clearly see, e.g., which combinations of qualifying questions you have covered and which you have intentionally omitted (pipe delimiters are not special/required).

We suggest keeping the parameters in the flow order on the first screen for convenience. For the same purpose, elements like this can serve as visual dividers on the Parameters/Scenarios screens and are optional:

Once the draft model is created, the auto-generated Mind Map presents an intuitive view of the model elements and can be easily shared for collaboration and approval.

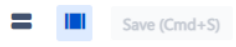


This approach allows teams to communicate clearly and collaborate efficiently by confirming early in the testing process that 1) all critical aspects have been accounted for, and 2) they have been included at the right level of detail (which is one part of the TCD answer to “how much testing is enough?”).

Building a Model – Step 2 – Generating Optimal Scenarios

> All constraints are for illustrative purposes and do not reflect the actual logic of the sample applications linked in the opening.

The horizontal model shape we mentioned earlier increases the benefits delivered by a relatively new TCD feature – multi-value constraints.




```
Strategy[Single Leg - Equity,Single Leg - Index] ≠ Transaction Type[N/A]
Strategy[Single Leg - Equity,Single Leg - Index] ≠ Order Type[Debit,Credit,Even,Market]
Strategy[Spread,Straddle,Strangle,Collar,Buy-Write,Married Put,Butterfly,Condor] ≠ Transaction T
Strategy[Spread,Straddle,Strangle,Collar,Buy-Write,Married Put,Butterfly,Condor] ≠ Order Type[L
```

They allow specifying the rules in a faster and more concise manner compared to the standard icon view (if you do not see the toggle in the top left of the Constraints screen, feel free to reach out to us and request beta access).

Business logic for parameters like User Profile often involves more than 2 factors, which may complicate the implementation of TCD constraints.

You can refer to the “How to Implement N-way Constraints” article for available methods, and we are working on improving that functionality in the future.

The last point at this step is the algorithm thoroughness selection. System-to-system TCD models typically utilize 2-way or Mixed-strength (depending on the parameter structure, project criticality, etc.). The dropdown settings in Mixed-strength are generally chosen based on the following parameter logic:

- Does it impact all systems and has numerous rules/dependencies associated with it? -> Include with **at least** 2-way coverage selection.
- Does it impact all systems and has few/no rules/dependencies associated with it? -> Include with 2-way selection given short value lists + value expansions.
- Does it impact only 1 system but has numerous rules/dependencies associated with it? -> Include with 1-way coverage selection and a fairly exhaustive list of values (because of the constraints).
- Does it impact only 1 system and has few/no rules/dependencies associated with it? -> Likely should not have been included in the model, but 1-way otherwise. Any mandatory informational parameters would be in this category.

For this example, we will assume that User Profile (System A “outcome”), Account Type (System B “outcome”), and Order Type are critical for the release and deserve 3-way, the rest of the model stays at 2-way because we only included the important variables.

The resulting scenarios table could look like this:

2-way Substantial loss reaction	3-way System A v2 outc... Profile	2-way System B	3-way Account Type	3-way Order Type	2-way Initial Investment
immediately switch	1	******	Self-service Brokerage Account	Limit	0
wait three months	2	******	Dedicated Financial Advisor	Stop / Limit	5000 - 5000
wait one year	3	******	Mix	Market	100000
stay with the ...trategy	4	******	Mix	CAB	150000 - > 100000
stay with the ...trategy	5	******	Dedicated Financial Advisor	Market on close	2000000 - > 100000
immediately switch	6	******	Self-service Brokerage Account	Debit	100000
wait one year	5	******	Dedicated Financial Advisor	Even	0

The ability to iterate at this step and quickly regenerate test cases (based on, e.g., requirement updates) for review provides immense help clarifying ambiguities much earlier in the process.

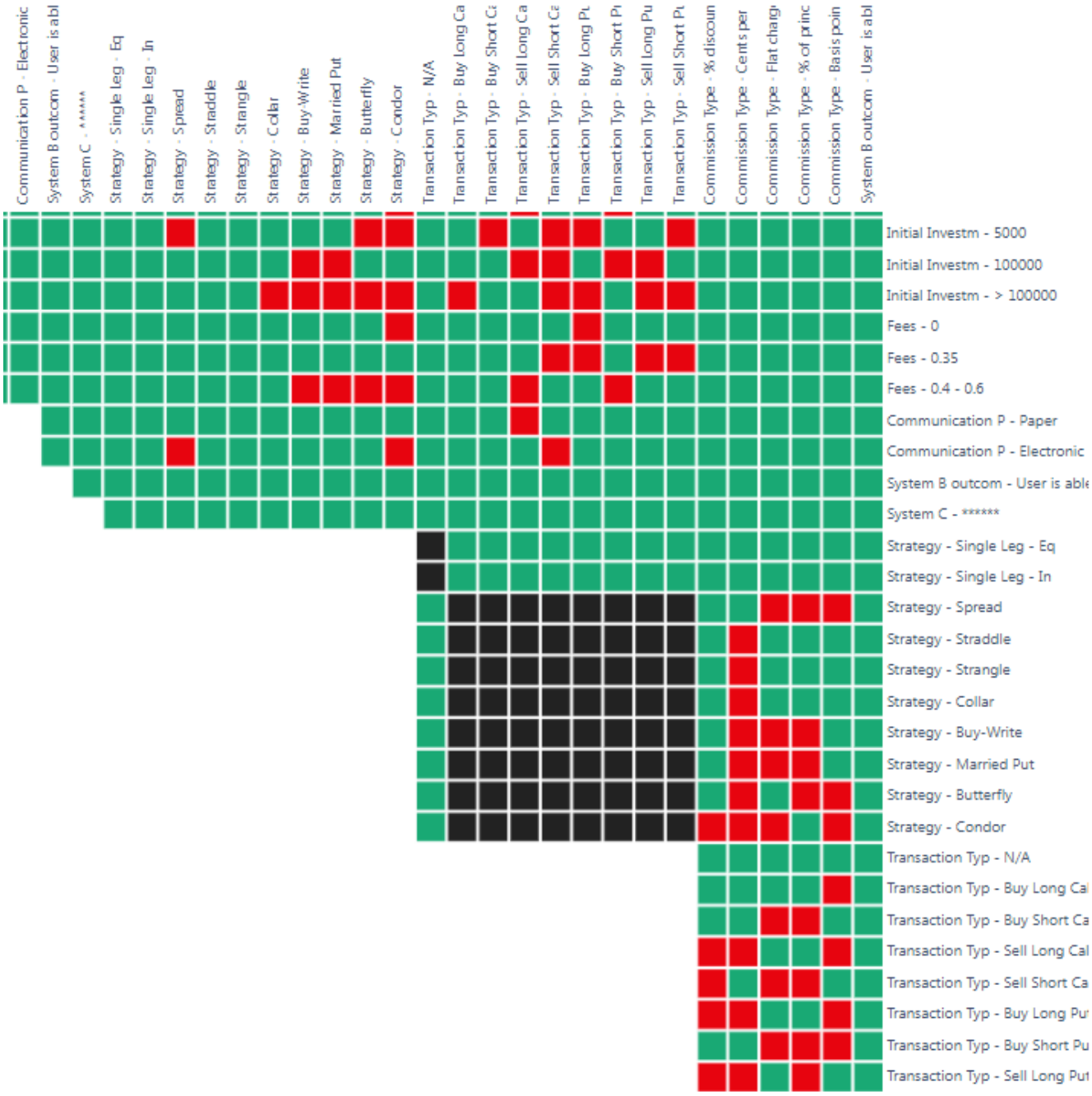
The effective combination of the level of detail in Parameters and the business-relevant coverage strength in Scenarios guarantees that the Test Case Designer algorithm optimizes your total model scope to have a minimal number of tests that cover all the important interactions.

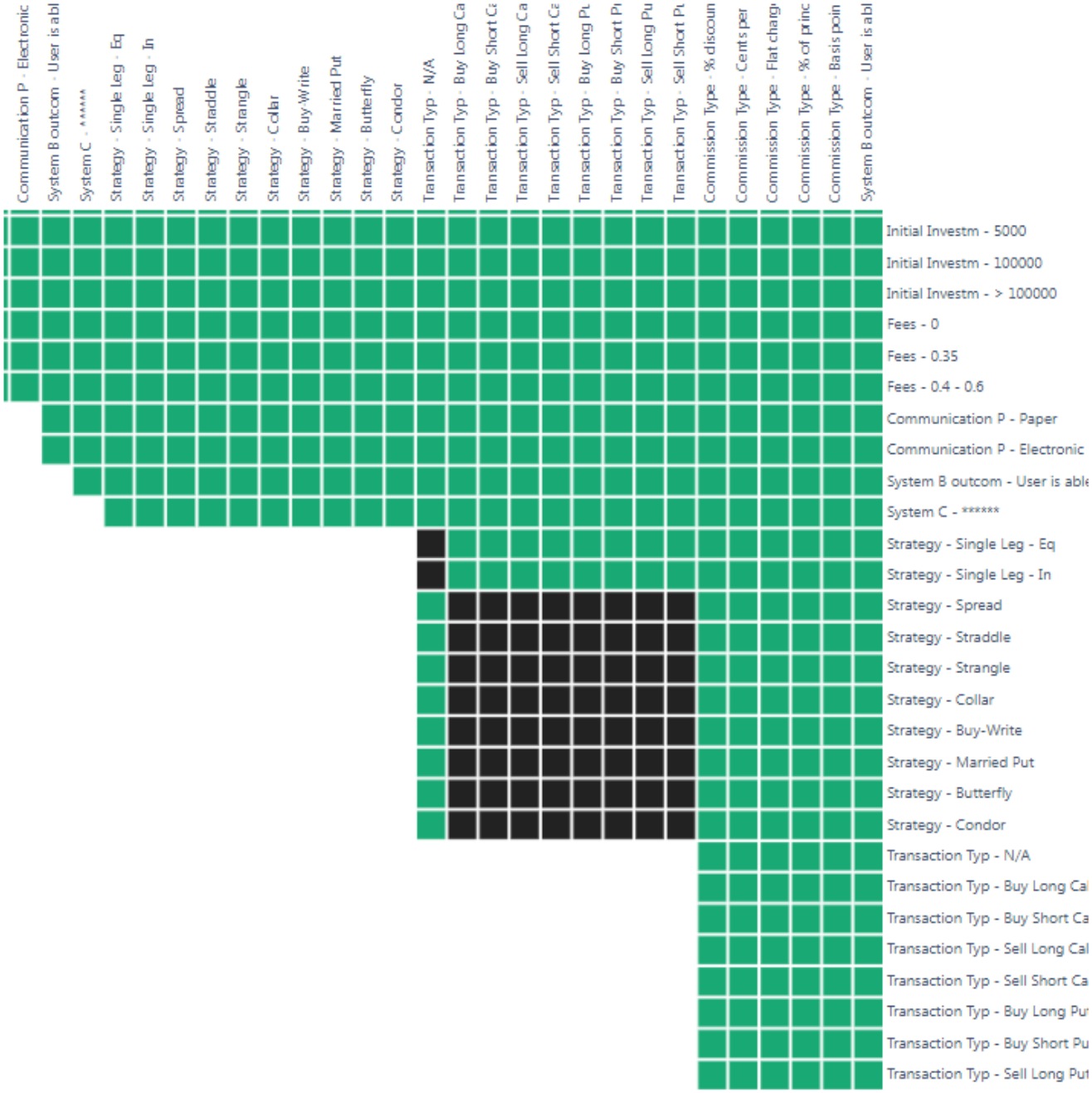
And next, we will discuss the last piece of the core testing “puzzle” – given the total scope, how we can use Test Case Designer visualizations to select the right stopping point.

Building a Model – Step 3 – Coverage Results Comparison

When integration scenarios are created by hand, they often represent a very fragmented view of the system and struggle with redundancy or omissions. Instead, TCD maximizes the interaction coverage in fewer scenarios and provides complete control and traceability for each of the steps in the test case.

If we now analyze the interaction coverage achieved and compare it with the typical manual solution, the results would often look like this:


Communication P - Electronic
System B outcom - User is abl
System C - ******
Strategy - Single Leg - Eq
Strategy - Single Leg - In
Strategy - Spread
Strategy - Straddle
Strategy - Strangle
Strategy - Collar
Strategy - Buy-Write
Strategy - Married Put
Strategy - Butterfly
Strategy - Condor
Transaction Typ - N/A
Transaction Typ - Buy Long Ca
Transaction Typ - Buy Short Ca
Transaction Typ - Sell Long Ca
Transaction Typ - Sell Short Ca
Transaction Typ - Buy Long Pu
Transaction Typ - Buy Short Pu
Transaction Typ - Sell Long Pu
Transaction Typ - Sell Short Pu
Commission Type - % discoun
Commission Type - Cents per
Commission Type - Flat charg
Commission Type - % of princ
Commission Type - Basis poin
System B outcom - User is abl
Initial Investm - 5000
Initial Investm - 100000
Initial Investm - > 100000
Fees - 0
Fees - 0.35
Fees - 0.4 - 0.6
Communication P - Paper
Communication P - Electronic
System B outcom - User is able
System C - ******
Strategy - Single Leg - Eq
Strategy - Single Leg - In
Strategy - Spread
Strategy - Straddle
Strategy - Strangle
Strategy - Collar
Strategy - Buy-Write
Strategy - Married Put
Strategy - Butterfly
Strategy - Condor
Transaction Typ - N/A
Transaction Typ - Buy Long Cal
Transaction Typ - Buy Short Ca
Transaction Typ - Sell Long Cal
Transaction Typ - Sell Short Ca
Transaction Typ - Buy Long Put
Transaction Typ - Buy Short Pu
Transaction Typ - Sell Long Put



As you can see, TCD-generated tests benefit from Intelligent Augmentation that ensures coverage of both (1) all specified requirements and (2) every critical system interaction. Our scenarios consistently find more defects than hand-selected test sets because interactions are a major source of system issues.

Taking this analysis a step further, given typical schedule deadlines, etc., we can identify the exact subset of the total scope that will be sufficient for the immediate testing goals and communicate that decision clearly to the management with the combination of the Mind Map + Coverage Matrix.

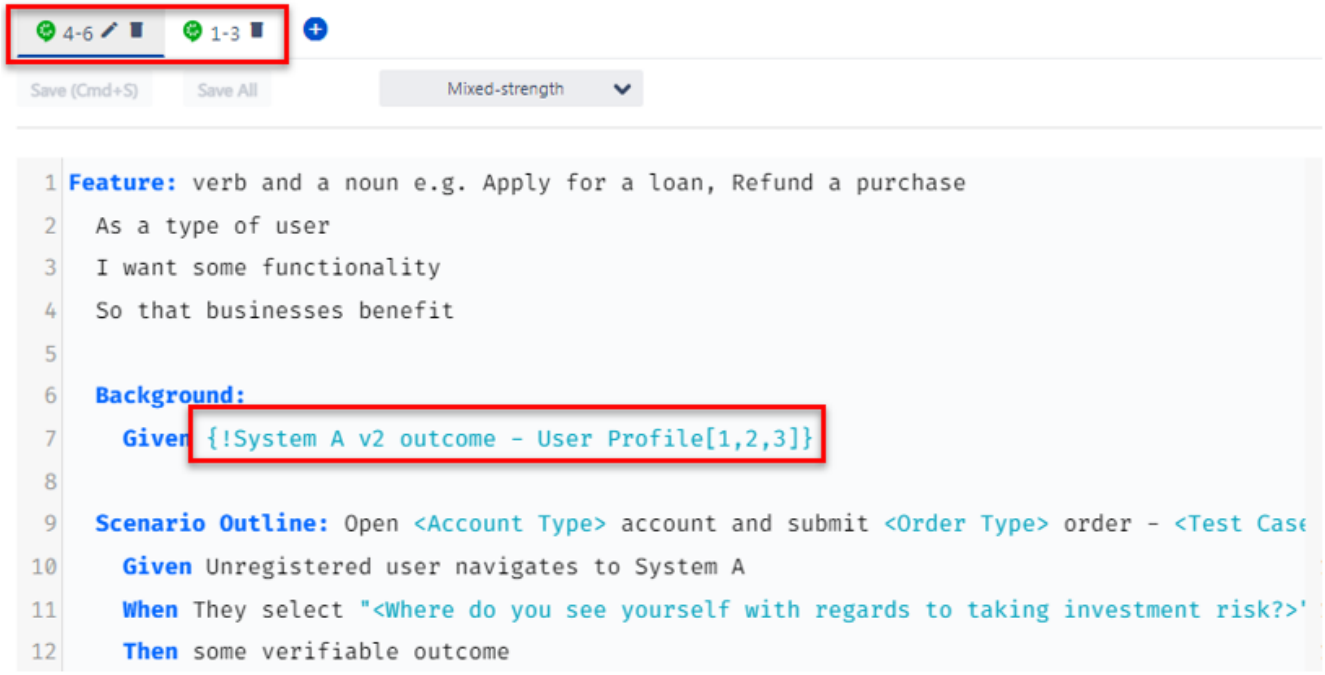
Building a Model – Step 4 – Scripting & Export

> Some teams may choose to execute directly from the test cases table. They would leverage “Save As” dropdown on the Scenarios screen and skip this section.

We are observing more & more teams switching to BDD, so we will be covering TCD Automate in this article, but most general principles also apply to Manual Scripts.

First, the overall script structure is completely up to you. The number of steps, length of each, number of parameters per each, etc. depend on your guidelines for both test design and execution – Test Case Designer **has the flexibility to support a wide range of preferences**.

Second, for the review and export efficiency, we will be using multi-value {[]} filters to separate User Profile 1-3 and 4-6 scenarios (assuming they have different validation steps for example purposes).



> "!" sign means negation of the values in the filter. You can check the "Usage" button on the Automate screen for more details about the syntax rules.

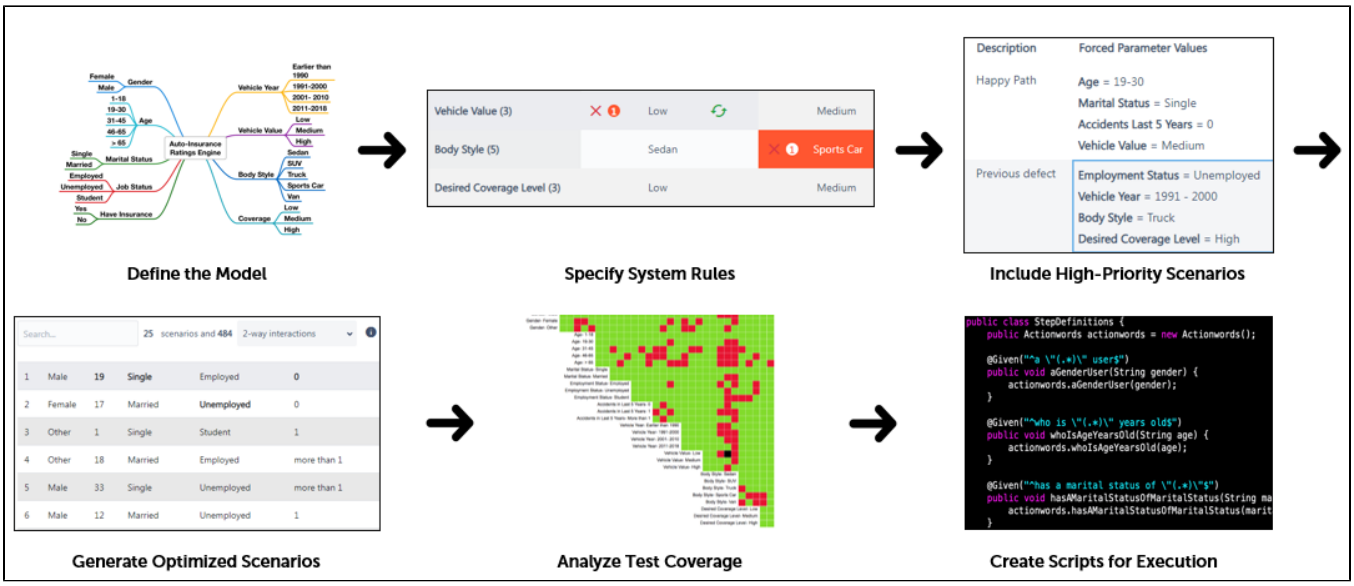
Lastly, we strongly recommend sharing the TCD models with automation engineers early in the process to allow them enough time to review and provide feedback on step wording, value names, etc.

Once the script is finalized, you can proceed to export the scenarios in the format compatible with the test management tools and/or automation frameworks. E.g., without any extra actions, you can generate the CSV for Xray alongside Java files.

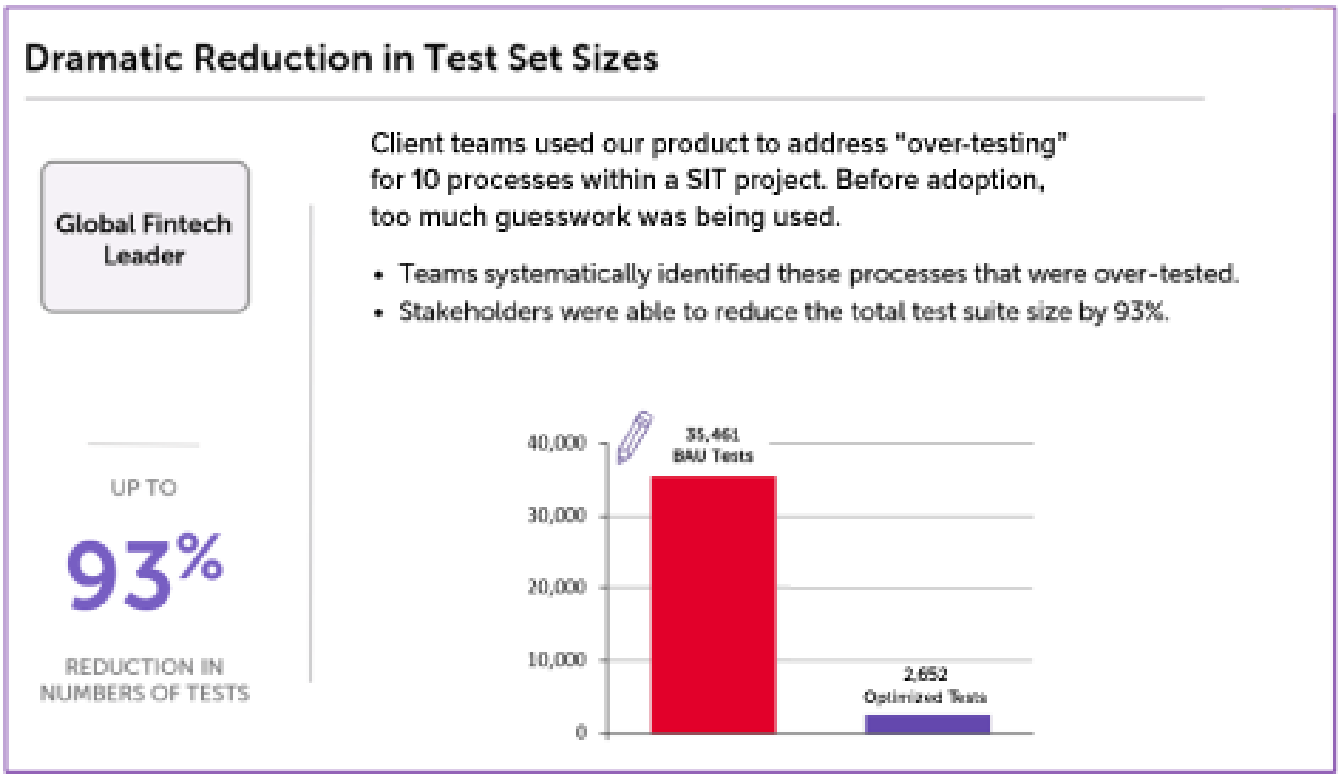
This step enables accelerated, optimized automation because you can:

- Rapidly create clear, consistent steps that leverage Behavior Driven Development principles.
- Export 1 Scenario block into multiple scripts based on the references to the data table.
- Improve collaboration across business and technical teams to understand the testing scope as a group.

Summary & Case Studies



In conclusion, the combination of Test Case Designer features will allow you not only to quickly generate the optimal set of scenarios but also to answer the **"how much testing is enough?"** question with clarity & confidence.

The image above should be familiar from our other educational materials, and hopefully, it underscores the notion that the process & methodology are not strongly dependent on the type of testing, type of system, industry, etc.

The goal of applying TCD is to deal with such challenges of manual test creation as prolonged and error-prone scenario selection, gaps in test data coverage, tedious documentation, and excessive maintenance.