Test Design Optimisation

Interactive content – best viewed in Full-Screen mode

EY
Building a better working world

01

Test Design Optimisation

The process of quality assurance and testing consists of test cases that are based upon requirements or user stories. To achieve the required level of coverage a certain number of tests must be executed, but these test suites typically contain redundancy. Traditional approaches to testing have proven to be inefficient, time-wasting and extends deployment timelines. Where quality at speed is a key driver behind digital transformation, this is unacceptable. It isn't a secret that due to resource and time constraints, test creation and execution are usually squeezed, resulting in the validity of test coverage to become questionable.

Traditional approaches add further risk to projects where defects are found in production. The impact of a severe production defect can be catastrophic. How should organisations change the approach? What are the practical solutions to address these challenges? EY recommend Test Design Optimisation through pairwise techniques.

THE BENEFITS OF IMPLEMENTING TEST DESIGN OPTIMISATION ARE:

01 Significant (up to 50%) cost reduction for testing

02 Increased number of defects found earlier in the lifecycle

03 Targeted test coverage with clear visual representation for both business and technical teams

04 Automatic design of high-quality 'multiple strike' test cases

05 Optimisation of test case volumes

06 Re-focus Regression Testing and Test Automation to be risk-driven

07 Enables full re-use of tests along the entire testing life cycle

02

Pairwise-based testing

Pairwise is a recognised testing technique that has been around for several years. Digital transformation and the demand to increase speed to market has driven a refocus on tools and automation to perform testing faster while not compromising quality. Historically, pairwise tools were only used on stable systems; however, applying them as part of the 'shift-left' approach while building the system has shown it can considerably reduce the cost, time and effort for testing. The industry defines pairwise testing (also known as all-pairs testing) to be a combinatorial method of software testing that, for each pair of input parameters to a system (typically, a software algorithm), tests all possible discrete combinations of those parameters. Using carefully chosen test vectors, this can be done much faster than an exhaustive search of all combinations of all parameters, by 'parallelizing' the tests of parameter pairs.

Pairwise is used to achieve the desired result in the most efficient way possible. When given a set of input parameters, the algorithm derives a minimum number of scenarios with all discrete value pairs to be tested at least once. When testing a complex application, there can be an enormous number of possible tests that could potentially be executed. There is usually no clear mathematical basis to identify the test coverage and no visual analysis of all the possible tests in one single location, mostly because documentation is either not updated with the latest requirements or is scattered across systems. With information scattered across multiple sources, there is a repeated effort to test the same or a similar test scenario. This creates the wastage and redundancy in the testing lifecycle. Pairwise tools not only provide optimized data combinations but also have an analysis capability that visually shows the pairs and coverage percentage and allows you to adjust to view the impact.

The increasing demand for speed to market puts more pressure on testing to be able to handle changes and new requirements faster. This demand presents other challenges as well; with reduced timelines the ability to test everything in a shorter timescale becomes a real issue. Additionally, to protect quality and reduce risk the right level of depth and test coverage for the highest-risk areas is vitally important. This feeds the test automation selection and therefore needs to have high focus.

Traditional approaches are very time-consuming and have limited ability to optimize. Implementing Test Design Optimization through a purpose-built tool will reduce the time required to define all the combinations and generate the test cases.

03

The Pairwise Tool

Without specific tools, it can be extremely challenging for testers and managers to determine how much testing is enough. A pairwise test tool enables the creation of the optimal mix of scenarios and guides teams to define and schedule risk-based testing. Pairwise tools provide modelling capabilities that are easy to undesstand and simple to use. Furthermore, it is easy to maintain test cases by adding new features or eliminate those not needed.

Pairwise is used to achieve the desired result in the most efficient way possible. When given a set of input parameters, the algorithm derives a minimum number of scenarios with all discrete value pairs to be tested at least once. When testing a complex application, there can be an enormous number of possible tests that could potentially be executed. There is usually no clear mathematical basis to identify the test coverage and no visual analysis of all the possible tests in one single location, mostly because documentation is either not updated with the latest requirements or is scattered across systems. With information scattered across multiple sources, there is a repeated effort to test the same or a similar test scenario. This creates the wastage and redundancy in the testing lifecycle. Pairwise tools not only provide optimized data combinations but also have an analysis capability that visually shows the pairs and coverage percentage and allows you to adjust to view the impact.

WHEN TO APPLY PAIRWISE TESTING

- Projects with a significant testing effort, where test design efficiencies and coverage improvements will have the most impact
- First-release projects or projects in the early stages of test creation; this will allow for earlier defect detection and for test efficiencies to be realized over a longer period of time
- Projects that require multiple data combination to be tested, resulting in a large number of test cases
- Web-based applications with significant decision-making processes
- Part of a Test Automation Transformation initiative where the entire test process that enables automation is optimised by selecting automation candidates through a pairwise driven approach

04

Pairwise-based testing

Pairwise can be used with the two leading design approaches:

- Model-based testing concept that records requirements and then aids in designing test scenarios and test cases
- Data parameter and value-driven

The tools identified in this paper serve as examples for the use of pairwise with these differing approaches.

Process model tools typically feature add-ons that allow the recording of out-of-the-box scripts with a User-Interface recorder tool (e.g. Selenium) and then import them to generate initial process models. Model-based tools are most effective when deployed at the beginning of the project, when requirements are being formed.

Data parameter and values tools are best applied when test scenarios are existing and the parameters needed to test with are defined by rules or code logic.

Following the input of process models or parameters and values the next step is to create any constraints and specific requirements if required. The tool then will automatically generate the test cases after which the tester can update any test cases and then adjust the degree of coverage. Coverage metrics provide a graphical view of the test case coverage level for the test cases selected.

A PAIRWISE EXAMPLE

An example use case for pairwise testing is 'Know your Customer (KYC)'. KYC requires a list of questions to be asked to verify whether a customer is eligible for banking products. Based upon answers provided, questions render additional questions and therefore can take multiple paths. Pairwise is an excellent technique to validate that all questions are tested. It takes those questions as parameters and the answers as values; constraints will be entered, then the tool will generate a set of test cases that will test the minimum coverage of all paths of parameters and values.

– 05 –

TEST DESIGN TOOLS

> > CLICK HERE TO VIEW TEST DESIGN TOOLS < <

06

Summary

Where quality at speed and system resilience are critical to digital transformation the quality assurance and testing process must be optimised to enable this. Test Design Optimisation in the form of pairwise is not industry-specific and therefore can be used across multiple industries and applications. Successfully implementing techniques and tools for pairwise have proven to reduce cost, time and effort across the entire testing lifecycle. Defect to test case ratios are dramatically lower when using pairwise. Implementing pairwise as part of a 'shift-left' testing approach leads to a reduction in cost of defects. Implementing a risk-based strategy underpinned by pairwise not only leads to accelerated delivery but also establishes a risk-based focused to ongoing systems regression testing. Converging this approach with Test Automation Transformation will position organisations testing process ready for the predicted increase of digital transformation initiatives.

If you would like to find out more or arrange a demo, please contact:

Matthew Steer
Associate Partner
CESA Advisory | Technology | Strategic Testing Services
Matthew.Steer@pl.ey.com
+48 660 440 119

Daniel Wilenski
Senior Consultant
CESA Advisory | Technology | Strategic Testing Services
Daniel.Wilenski@pl.ey.com
+48 789 407 567

EY | Assurance | Tax | Transactions | Advisory

About EY

EY is a global leader in assurance, tax, transaction and advisory services. The insights and quality services we deliver help build trust and confidence in the capital markets and in economies the world over. We develop outstanding leaders who team to deliver on our promises to all of our stakeholders. In so doing, we play a critical role in building a better working world for our people, for our clients and for our communities.

EY refers to the global organization and/or one or more of the member firms of Ernst & Young Global Limited, each of which is a separate legal entity. Ernst & Young Global Limited, a UK company limited by guarantee, does not provide services to clients. Information about how EY collects and uses personal data and a description of the rights individuals have under data protection legislation are available via ey.com/pl/pl/home/privacy.

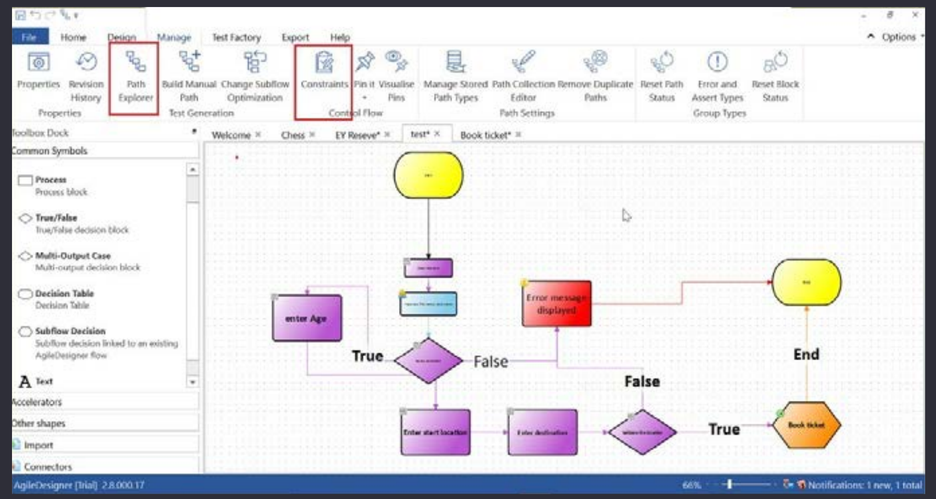
For more information about our organization, please visit ey.com.

Test Design Tools (1/4)

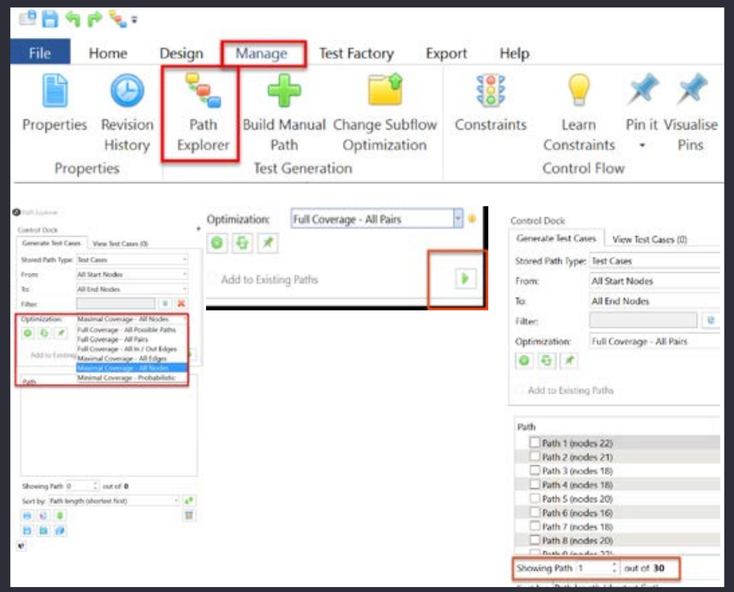
<< GO BACK TO MAIN PAGE

MODEL-BASED TEST DESIGN TOOL

The tool used in this example is a model-based pairwise tool. The models can be built manually by dragging and dropping symbols based on business process flows, or they can be uploaded from an existing process flow tool. This tool has an add-on that works with Selenium that will allow you to record application steps that can be imported to create the process models. Each of the symbols represents a different step at which constraints, values and specific requirements can be applied.



The next step is to generate the test cases. Select the type of optimization you want to perform and then generate the test cases. All paths and the number of nodes will appear for selection and review. Save the paths generated and then the test coverage for each path can be visualised and confirmed.



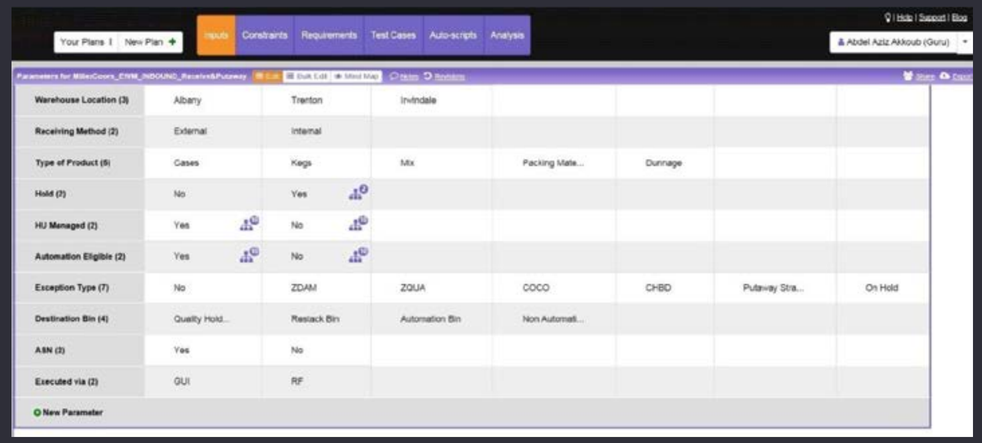
Test Design Tools (2/4)

<< GO BACK TO MAIN PAGE

DATA PARAMETER-DRIVEN TEST DESIGN TOOL

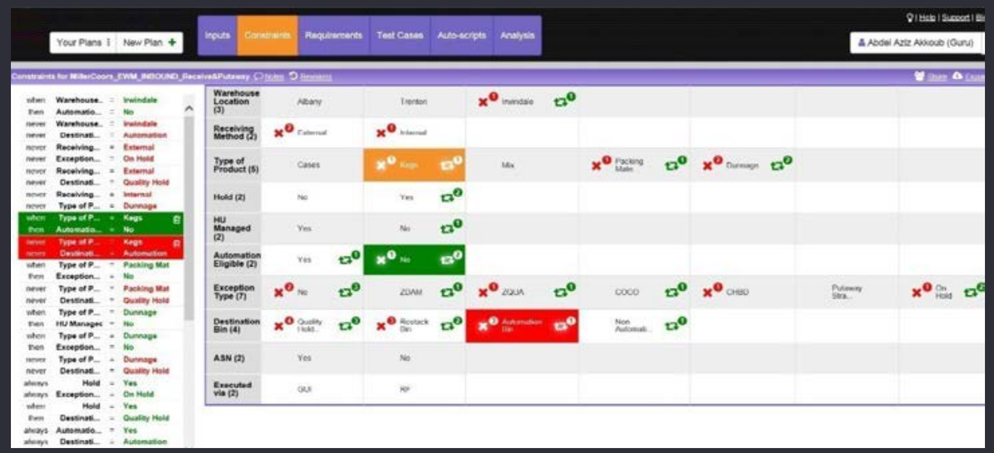
Input

This example uses a tools to enable pairwise testing. The test planner creates a new plan or can select to update an existing plan. Once in the plan, the Input tab allows the test planner to enter parameters and values required for testing. Parameters and values are the indicators that change from test case to test case. Values are the different selections for a parameter. The more parameters entered for a test plan, the less effective the tool becomes and therefore systemic planning and creation of multiple plan modules should be considered.



Constraints

After parameters have been entered the next step is to enter any constraints against those parameters. Constraints are rules that define how the parameters and their values should work with other parameters and values. For example, a parameter and one of its values may only be allowed when another parameter and its value are met (invalid pairs, married pairs). If/then statements also help in defining constraints. Constraints can be 1:Many or Many:1.



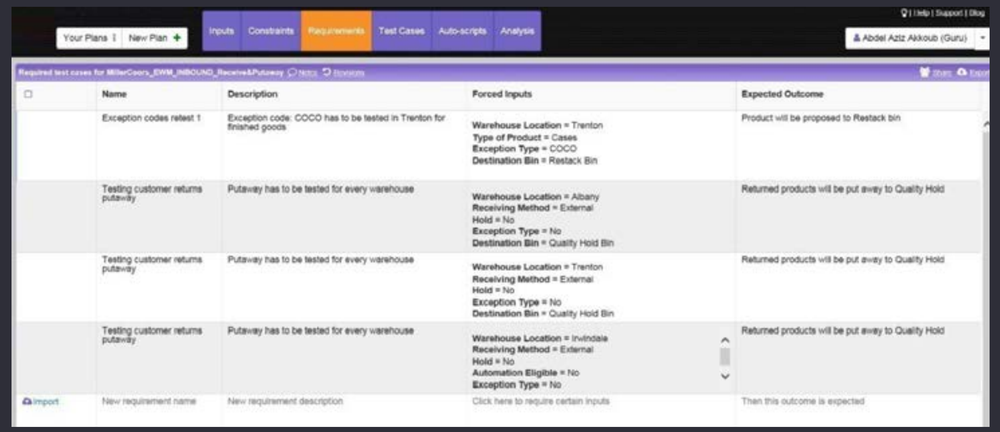
Test Design Tools (3/4)

<< GO BACK TO MAIN PAGE

DATA PARAMETER-DRIVEN TEST DESIGN TOOL

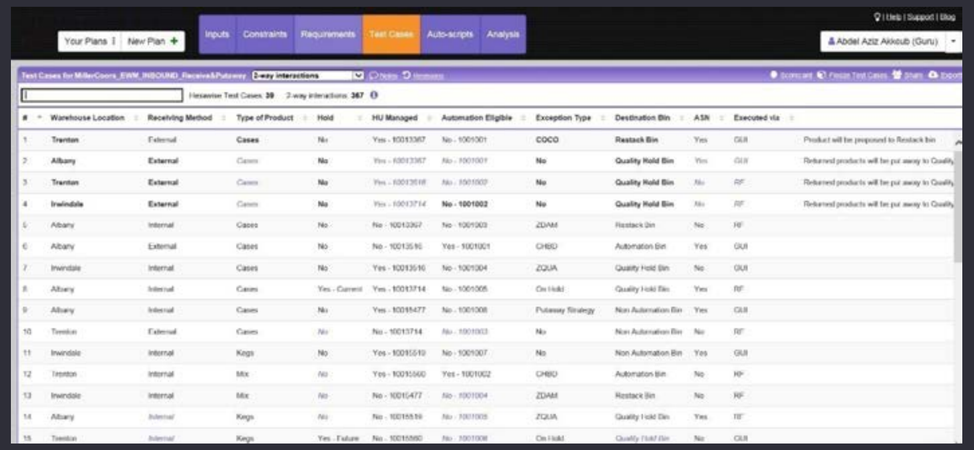
Requirements

Requirements can be imported from several formats, such as Word or Excel. When using Excel, data inputs will need to be in the format of Name, Description and Expected Outcome. After importing requirements, specific combinations or test conditions/test inputs can be added. Traceability is enabled from requirements to parameters. The Requirements tab allows the test planner to manually input a set married pair, constraints or a set of multiple combinations of data in case not automatically created by the tool. Negative tests are also created and can be manually inputted to cover error handling cases.



Test Cases

The test cases are generated. The default setting for test cases is two-way interaction, but can be configured to select up to a five-way interaction or mixed-strength interaction. Increasing the number of interactions increases coverage and reduces risk. However, as you increase the number of interactions, the number of test cases increases and, in return, time and effort increase. Risk is the driving factor behind the test coverage at this point and should be agreed by business and technical stakeholders.



Test Design Tools (4/4)

<< GO BACK TO MAIN PAGE

Analysis

The Analysis tab illustrates the test coverage in different formats such as Plan Scorecard, Coverage Graph and Matrix Chart. These can be used to determine how much test coverage has occurred at any time and how you can maximize investment on return. The Coverage Graph shows that as you increase the number of tests, the percentage of coverage starts to reach a point that creates a negative return. The Matrix Chart shows what is being covered based on the number of tests you select to perform. The Plan Scorecard lists the features used and shows what is tested. The Analysis tab demonstrates empirically that a scope coverage of approx. 80% can usually be obtained by executing less than half of the tests. The Matrix Chart also provides a road map to constructing negative tests, as it illustrates what should not be occurring within the application under test.