How to Integrate Test Case Designer with Robot Framework



References

In the "How to Optimize Data-driven Automation with Test Case Designer" tutorial (Server/DC ; Cloud), w e made a quick note that the efficient script-based approach is also possible in Test Case Designer (TCD). So, we wanted to explore that topic more and give the spotlight to integrating TCD with Robot Framework.

Robot Framework is an open source test automation framework for acceptance testing and acceptance test-driven development. It follows different test case styles keyword-driven, behaviour-driven and datadriven for writing test cases.

Some of the key Robot Framework advantages include ease of installation and use, good support for both external libraries and built-in/custom keywords, and more readable & maintainable test cases.

Overview

One of the most common concerns we hear from clients is:

"Our current tests are good enough. Why invest the time and resources in adopting the Test Case Designer methodology if it doesn't move the needle?"

That may be true, but we don't have to guess - TCD Analysis capabilities allow us to make a pretty accurate comparison (you can learn more in the "How Are TCD Tests Objectively Superior?" tutorial from the "Why is Test Case Designer helpful?" section). For this tutorial, we will use 2 test suites created for **th e same requirement/coverage goal**:

- "TCDBookingTests", representing the optimized output from TCD with 15 tests;
- "TypicalBookingTests", representing the manually designed suite with 33 tests (based on our experience, it shows an example of a "pretty good job" for the manual effort type).

The requirement is to provide hotel availability options for the given reservation details at https://www.trip. com/. We check this by validating that clicking the "Search" button successfully leads to the "Results" page (by "properties found" page content).



We will first describe the integration steps for "TCDBookingTests" and then compare the coverage and execution metrics between the two.

Prerequisites

In order to run this tutorial you need to have Python and Robot Framework installed.

Integrating Test Case Designer with Robot Framework

Build a TCD Model

destination (2)	location 👗 \varTheta	landmark/property/etc. 📕 🔕				
checkindate (3)	15	24	31			
checkoutmonth (2)	May 2023	Jun 2023				
checkoutdate (4)	3	12	30	31		
rooms (3)	1	2	3			
adults (4)	1	2	3	4		
children (3)	0	1	2			
workcheckbox (2)	checks	ignores				
rating (6)	00	2	3	4	5	3 or 4

It is built using a fairly straightforward approach of a parameter per UI field with a couple things to note:

- For the Destination field, we are more interested in classes of search terms, less in the syntax rules. So, we use 2 Values to represent categories, then add specific search terms using Value Expansions.
- The date-related parameters will need to be updated within the model to maintain execution accuracy. The screenshot represents the model state at the time of publication. Alternatively, you can consider using abstract values in TCD ("this month", "next month", "last day of the month", etc.), then let automation determine the specifics.
- For conditional steps, Robot Framework allows to use any syntax as the trigger, so we can
 adjust the value naming to match the context "checks" for the checkbox, numbers for the rating
 stars. Neither wording has any special function within TCD itself.

Constraints should account for the date logic and the room-vs-people limitations:

```
1 checkoutmonth[May 2023] ≠ checkindate[31]
2 checkoutmonth[May 2023] ≠ checkoutdate[3, 12]
3 checkoutmonth[Jun 2023] ≠ checkoutdate[31]
4 rooms[2] ≠ adults[1]
5 rooms[3] ≠ adults[1, 2]
```

For scenario generation, we assume the exact checkout date and the ratings are not as important, so we switch them to 1-way:

Search	15 scenarios and	159 Mixed-st	rength interactions	- 0					🔒 Freeze
Reapply	2-way ♥ destination	2-way 👻	2-way •	1-way • checkoutdate	2-way ¥	2-way ♥ adults	2-way ♥ children	2-way • workcheckbox	1-way 🕶 🗧
1	Myrtle Beach - location	15	Jun 2023	3	1	3	0	checks	no
2	Statue of Libety/etc.	24	May 2023	30	2	4	1	ignores	2
3	CIT - landmark/oronartu/atr	31	hun 2023	12	1	2	2	innores	3

You can import the model and try this process yourselves:



Create a Robot script

The key advantage at this step is that a user creates & maintains a single data-driven template in TCD Automate and exports as many tests as there are rows on the TCD Scenarios screen, with all the parameter values replaced inline. That combines efficiency and clarity/ease of use.

TCD Automate Script *** Settings *** A test suite with TCD RF export in Gherkin style. TCD Documentation Automate allows to create & maintain 1 data-driven template while generating many Robot test cases with in-line variable values. This file contains the risk-based (mixed-strength) TCD suite (15 TCs) with lower priority on checkout date and rating Library BuiltIn Library Dialogs SeleniumLibrary Library *** Variables *** \${BROWSER} headlesschrome \${DELAY} 1 \${START URL} https://www.trip.com/ \${DESTINATION TXTFIELD} //*[@id="hotelsdestination"] \${CHECKIN BTN} //*[@id=" searchBoxCon"]/div/div/ul/li[2]/div/div[1] \${ROOMSPLUS BTN} //*[@id="searchBoxCon"]/div /div/ul/li[3]/div/div[3]/div[1]/div/span[3] \${ADULTSPLUS BTN} //*[@id="searchBoxCon"] /div/div/ul/li[3]/div/div[3]/div[2]/div/span[3] \${CHILDRENPLUS BTN} //*[@id="searchBoxCon"] /div/div/ul/li[3]/div/div[3]/div[3]/div/span[3] \${CHILDRENNAGE DIV} //*[@id="searchBoxCon"] /div/div/ul/li[3]/div/div[3]/div[4]/div \${CHILD1AGE DROPDOWN} //*[@id="searchBoxCon"]/div /div/ul/li[3]/div/div[3]/div[4]/div[1]/select \${CHILD2AGE DROPDOWN} //*[@id="searchBoxCon"]/div /div/ul/li[3]/div/div[3]/div[4]/div[2]/select //*[@id="searchBoxCon"] \${DONE1 BTN} /div/div/ul/li[3]/div/div[3]/div[4]/span \${DONE2 BTN} //*[@id="searchBoxCon"] /div/div/ul/li[3]/div/div[3]/div[5]/span \${WORKTRAVEL CHECKBOX} //*[@id="searchBoxCon"]/div /div/div/div[1]/div[1] \${SEARCH BTN} //*[@id=" searchBoxCon"]/div/div/ul/li[5]/div *** Test Cases *** Valid Booking Requests_TC<Test Case>_for <destination> with <rooms> rooms and <adults> adults TAC-242 [Tags] Given browser is opened to start page When user selects destination "<destination>" And user selects May 2023 "<checkindate>" check-in date and "<checkoutmonth>" "<checkoutdate>" check-out date And user selects "<rooms>" rooms for "<adults>" adults and "<children>" children And user "<workcheckbox>" work checkbox and selects "<rating>" star rating Then clicking "Search" button generates results successfully *** Keywords *** Browser is opened to start page Open Browser \${START URL} \${BROWSER} options=add_experimental_option("excludeSwitches", ["enable-logging"]) Set Window Size 1920 1080 \${DELAY} Set Selenium Speed Page Should Contain Your Trip Starts Here

```
user selects destination "${destination}"
 Input Text ${DESTINATION TXTFIELD}
${destination}
user selects May 2023 "${checkindate}" check-in date and
"${checkoutmonth}" "${checkoutdate}" check-out date
       Click Element ${CHECKIN BTN}
      Click Element //div[@class = 'c-calendar-month title' and
text() = 'May 2023']/following-sibling::div//span[@class = 'day' and text()
= '${checkindate}']
                       //div[@class = 'c-calendar-month__title' and
      Click Element
text() = '${checkoutmonth}']/following-sibling::div//span[@class = 'day'
and text() = '${checkoutdate}']
user selects "${rooms}" rooms for "${adults}" adults and "${children}"
children
      Sleep 2s
Run Keyword If '${rooms}' == '2'
                                                      Click
Element ${ROOMSPLUS BTN}
 Run Keyword If '${rooms}' == '3'
                                                      Double
Click Element ${ROOMSPLUS BTN}
                         '${adults}' == '3' and '${rooms}' in
      Run Keyword If
['1','2'] Click Element ${ADULTSPLUS BTN}
Run Keyword If '${adults}' == '4' and '${rooms}' in
['1','2'] Double Click Element ${ADULTSPLUS E
                     Double Click Element ${ADULTSPLUS BTN}
      Run Keyword If '${adults}' == '4' and '${rooms}' ==
Click Element ${ADULTSPLUS BTN}
'3'
    Run Keyword If '${children}' == '1' Click
Element ${CHILDRENPLUS BTN}
  Run Keyword If '${children}' == '1' Select From List
By Value ${CHILD1AGE DROPDOWN} 12
     Run Keyword If '${children}' == '2'
                                                 Double Click
Element ${CHILDRENPLUS BTN}
 Run Keyword If '${children}' == '2'
                                                 Select From List
By Value ${CHILD1AGE DROPDOWN} <1
   Run Keyword If '${children}' == '2'
                                                  Select From List
By Value ${CHILD2AGE DROPDOWN}
                                      17
                         '${children}' == '0'
      Run Keyword If
                                                 Click
Element ${DONE1 BTN}
  Run Keyword If
                          '${children}' != '0' Click
Element ${DONE2 BTN}
user "${workcheckbox}" work checkbox and selects "${rating}" star rating
     Run Keyword If '${workcheckbox}' ==
'checks'
                          Click Element
                                             ${WORKTRAVEL CHECKBOX}
  Run Keyword If '${rating}' in ['2', '3', '4',
'5']
          Click Element //*[@id="searchBoxCon"]/div/div
/div/div[2]/span[${rating}]
                          '${rating}' == '3 or
     Run Keyword If
4 '
                                   Click Element
                                                      //*[@id="
searchBoxCon"]/div/div/div/div[2]/span[3]
   Run Keyword If '${rating}' == '3 or
4 '
                                   Click Element
                                                      //*[@id="
searchBoxCon"]/div/div/div[2]/span[4]
clicking "Search" button generates results successfully
      Click Element
                                     ${SEARCH BTN}
       Page Should Contain
                                     properties found
      Close Browser
```

```
Settings and Variables are defined similarly to any other Robot script. In the Test Cases section, there are a couple of points to highlight:
```

Test Cases section

```
*** Test Cases ***
Valid Booking Requests_TC<Test Case>_for <destination> with <rooms> rooms
and <adults> adults
    [Tags] TAC-242
    Given browser is opened to start page
    When user selects destination "<destination>"
        And user selects May 2023 "<checkindate>" check-in date and
"<checkoutmonth>" "<checkoutdate>" check-out date
        And user selects "<rooms>" rooms for "<adults>" adults and
"<children>" children
        And user "<workcheckbox>" work checkbox and selects "<rating>"
star rating
        Then clicking "Search" button generates results successfully
```

- TCD only supports the Gherkin style of Robot scripting.
- Requirement linking is handled via Tags.
- As in other TCD Scripting examples, the steps are parameterized in order to connect to the Scenarios table (e.g. "<destination>").

In the Keywords section, there are typically 3 ways of handling TCD parameters:

1. Direct argument

```
user selects destination "${destination}"
Input Text ${DESTINATION
TXTFIELD} ${destination}
```

2. Part of xpath

(ii)

```
Click Element //div[@class = 'c-calendar-month_title' and
text() = '${checkoutmonth}']/following-sibling::div//span[@class =
'day' and text() = '${checkoutdate}']
```

3. Part of condition (in this example, the age of children ("12") was not important enough to be a parameter/value expansion, so it's hardcoded directly in the script)

Run Keyword If	'\${children}' == '1'		Click
Element	\${CHILDRENPLUS BTN}		
Run Keyword If	'\${children}' == '1'		Select From List
By Value	\${CHILD1AGE DROPDOWN}	12	

Once the script is ready, we will export it into the Robot format:



You can choose to leverage your own IDE for the template writing (to benefit from IntelliSense, etc.) then copy it into TCD for the export process.

Execute and import results

We can run the tests from the command line, specifying the report name and format:

robot -d reports --output TCDDemoReport.xml TCDBookingTests.robot

If we choose to run them in parallel, we can install Pabot and utilize the following command:

pabot --testlevelsplit -d reports --output TCDDemoReport.xml TCDBookingTests.robot

Once the script completes the execution, we can import results using your CI/CD tool of choice, eventually with one of available plugins for them, or invoking Xray's REST API directly using `curl` utility. For example (parts in {} should be customized based on your details):

curl	-H	"Content-	•Type:	multipart/	form-data"	-u	{username}	}:{password	1} -F
"file	e=@r	eports/{R	leportN	Name}.xml"	"https://{	Jira	URL}/rest/	/raven/1.0,	/import
/exec	cuti	.on/junit?	projec	ctKey={Pro	jectKey}&te	estPl	anKey={Tes	stPlanIssue	eKey}″

(i) CI/CD

For the CI/CD process, please refer to this collection of tutorials - DC/Server ; Cloud

Also, you can learn more about Robot Framework integrations from these articles:

- Taking advantage of Robot XML reports (DC/Server)
- Testing using Robot Framework integration in Python or Java (DC/Server)
- Taking advantage of Robot XML reports (Cloud)
- Testing using Robot Framework integration in Python or Java (Cloud)

As a result, you should see 15 Test issues (linked to the requirement story and to the Test Plan issue) and a Test Execution issue (with the step-by-step breakdown under Execution details of each run):



One scenario in "TCDBookingTests" fails because the UI error tooltip appears about the maximum stay duration of 31 days. Before the selections, that limitation doesn't seem to be noted anywhere.

This could be an example of the execution feedback loop for combinatorial test design: we will often exercise the combinations that haven't been explicitly mentioned in requirements. If the 31-day limit hasn't been caught during the collaborative model creation, then, based on the execution failure, we would confirm the expected behavior and, if the error is valid,

- add the constraint to the TCD model between checkindate[15,24] and checkoutdate[30, 31] and /or
- replace the value lists to focus more on the valid range.

A few scenarios in "TypicalBookingTests" fail for the same reason.

Comparison and Conclusion

Here are the final versions of both suites:



Based on the corresponding execution reports, we can compare the time:

Total testing:	14 minutes 56.0 seconds
Elapsed time:	1 minute 11.73 seconds
Total testing:	26 minutes 50.9 seconds
Elapsed time:	1 minute 51.12 seconds

~15 minutes for the TCD suite vs ~27 minutes for the manually designed suite without parallelization (~40 seconds difference with parallelization). This doesn't account for the much faster time to create the TCD test suite.

With the help of TCD analysis capabilities, we can compare interaction coverage - **100%** for the TCD suite vs **84.9%** for the manually designed suite after 15 tests (96.8% for the manually designed suite after all 33 tests).

And, while harder to estimate, the increase in script **maintenance efficiency** (1 data-driven template in TCD vs individual scripts in typical approach) is important to keep in mind.

Coverage graph for the "Optimized" suite:

Q Parameters	Percentage of Interactions Tested Mixed-strength
> RULES	(Explain this analysis)
Scenarios	100
> SCRIPTS	90
V ANALYSIS	80
# Coverage Matrix	70
Coverage Graph	50
	40
> REVIEW	20
< Share	10
► Export	0 tests 2 tests 4 tests 6 tests 8 tests 10 tests 12 tests 14 tests

Coverage graph for the "Typical" suite. Notice that the horizontal axis:

1) is on a different scale, which makes the shape look more accelerated than the one above (while the actual percentages are lower on a test-by-test basis);

2) goes beyond 33 as TCD has to complete the mixed-strength coverage goal.

Q Parameters	Percentage of Interactions Tested Mixed-strength
> RULES	(Explain this analysis)
Scenarios	100
> SCRIPTS	90
V ANALYSIS	80
# Coverage Matrix	60
Coverage Graph	50
	40
> REVIEW	30 20
< Share	10
► Export	0 V 0 tests 4 tests 8 tests 12 tests 16 tests 20 tests 24 tests 28 tests 32 tests

Tips

- As mentioned above, this tutorial demonstrates the approach where "complete" Robot tests in Gherkin style are generated from TCD. A data-driven implementation similar to the Playwright /Cypress tutorial is also possible via a CSV/pipe-delimited data table exported from the TCD Scenarios screen + the Test Template in RF (or using a specialized library).
- For simplicity of the example, all executable code is in 1 file. You can split it into Resource.robot and Booking.robot, if desired.
- If the desired algorithm goal is 2-way, not mixed-strength, then Coverage Matrix can be used to review the specific pairwise gaps in addition to the aggregate coverage percentage.

References

- Taking advantage of Robot XML reports (DC/Server); Taking advantage of Robot XML reports (Cloud)
- How Are TCD Tests Objectively Superior?
- https://robotframework.org/SeleniumLibrary/
- https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html#differentoutput-files