Concept

GraphWalker

It is a very simple tool that is used to create models of how a system is supposed to function.
You can test a model with GraphWalker CLI, which is a test tool for the command line.
There is an offline mode and an online mode for CLI.

Offline - This mode can be used to run all steps in the model in one sequence, and then you can later use the steps for another program, ex. Selenium or RobotFramework. In other cases, you can just run them to prove that the model works as intended.

Online - This mode will be directly connected to a system under test (SUT) with either Websocket or HTTP Rest. This is the mode that I have used in this project.
This mode can be infinite, so that it never stops unless you turn off the system that is under test. It runs in parallel with the system under test.

RobotFramework

Is a test automation tool for acceptance testing. It can use pre built keywords to run in test cases. Uses Selenium library to be able to test websites.

Project idea

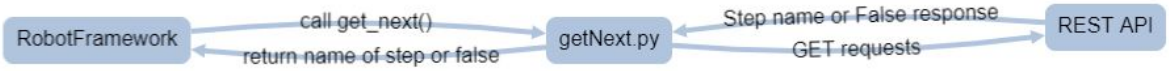
So the idea with this project is to use the simplicity of drawing a model of a system and also use the ability of running the system in a random way. Then to take that model and run the model steps in RobotFramework, which actually tests the functionality of the system.

What I have done so far

Here is a description of how the system works and a model of how the system works is below the description.

1. **RobotFramework** will make a call to a python script called **getNext.py**
2. **getNext.py** has a function which does a get request to the graphwalker **REST API online** to check if there are any more steps in the model (http://localhost:8887/graphwalker/**hasNext**)

3. The API will return **true** if there are more steps and **false** if there are no more steps (stop condition has reached it's limit)
4. If there are steps left, then it will do another get request to the API (http://localhost:8887/graphwalker/**getNext**). The **getNext** will get the name of the step, which will either be a **vertex** or an **edge**.
5. It will then **return** the name back to **RobotFramework**.
6. RobotFramework will now create a **new keyword** from the name and put it in the **test case**
7. If the keyword **exists** in a **keyword file**, then it will run the keyword.



What I have left for others to continue working on

GraphWalker Player - Daniel requested to have a visualization tool, which displays a model, like the one above, to show where the test currently is. There is something called GraphWalker Player, but it only works with Java and Websocket. So I took the index.html file that Kristian Karl (creator of GW) made, and began converting it to work with API calls from Flask server.

The idea is to load a model into the index.html which will be displayed with the help of **CytoscapeJS.**

I have managed to create a button that imports a model and displays it.
I have also created a Flask server that can receive and render the step name.

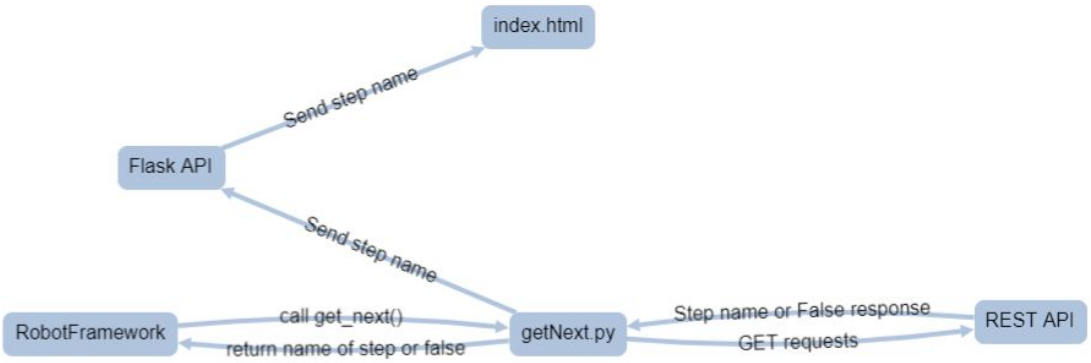
The **problem** is that the data must be displayed in the HTML-file without needing to refresh the page. The data disappears when you refresh the page and ofc the model is no longer visible.

Next step

So the next step is to make the name visible in the index.html file without the need to refresh the site.

AJAX might be a solution to the problem. So that's where I would look first.
When you can render the name without needing to refresh the page, then you can take the name and use it to start building the paths in the Cytoscape model.

Here is a picture of how the whole system should work.



If you have any questions, you can reach me on: ilarisilander@hotmail.com