System Under Test : Notepad Application

- Behavior of the SUT:

    - Initially, the Notepad application is not in running state.
    - Open the Notepad and create a file, edit a file and save or no save by exiting from the application.
    - Files are temporarily stored in a location in the format filename_nnnnn.txt [system generated]. Later, it can be retrieved if the user wants to append any information, or read its contents.
    - Test cases generated for the above mentioned scenario.

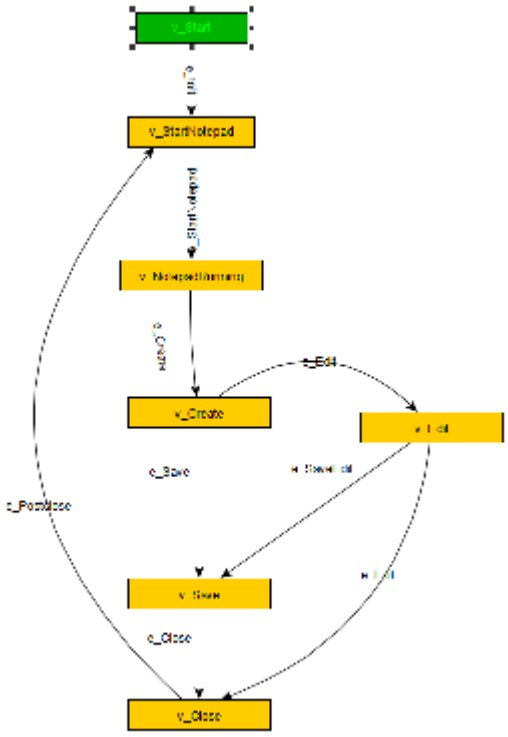
- The list of system inputs in your model with their description

INPUT	DESCRIPTION
Start Notepad	Notepad Application is stated and in running state
Create	Create a new File
Edit	Edit the existing file or new file created
Save	Save the newly created file or open an existing file and save
Exit	Exit the Notepad Application without saving after Edit state.
Close	Close the Notepad application after edit/save/no save.

- The transition table

STATE BEFORE	INPUT	STATE AFTER
Notepad not running	Start Notepad	Notepad is running
Notepad running and no file created.	Create File	Temporary file is created
Create File	Edit	File appended /formatted with new contents
File Open	Save	File is Saved
File Open	Exit	Edited file is closed without saving
File Open	Close	Exit the Notepad Application without saving the file.

- Graphical representation of the model in yED



- Stop-Criteria : The system stops testing once all the edges are covered to 100%.

  Generator: Random path execution

- Description of tests conducted:

  Scenario 1:

  Create and save file - User creates a file and saved. System checks whether the file created exists.

  Scenario 2:

  Edit and save file - Saved file is reopened and edited. System checks whether the new content is appended to the existing one and saved.

  Scenario 3:

Create and no save - A new file is created and exit the application. System checks if the temporarily created file has been deleted when the user has exited the application.

- Lessons learned from testing this particular SUT:

  - Understood the basic criteria of how to create a model and implementation of the same using Graphwalker.
  - Walk through the model with different generators and stop criteria.
  - How to implement auto-generated interface with the functionality of the model and generate test cases for the same.
  - Initially, time consumption was on the higher side to understand how to create a model and design it. Also, it might be little difficult for a new tester to understand the code flow and generate code / write test cases.
  - Once , the model has been designed and implemented seamlessly, it proved to be a efficient testing method.