Software Verification of Orion Cockpit Displays

M.A. Rafe Biswas, Samuel Garcia and Matthew Prado
Houston Engineering Center
University of Texas at Tyler
Houston, USA
mbiswas@uttyler.edu

Sadad Hossain, Matthew Souris, and Lee Morin
Rapid Prototyping Laboratory
NASA Johnson Space Center
Houston, USA

***Abstract*—NASA's latest spacecraft Orion is in the development process of taking humans deeper into space. Orion is equipped with three main displays to monitor and control the spacecraft. To ensure the software behind the glass displays operates without faults, rigorous testing is needed. To conduct such testing, the Rapid Prototyping Lab at NASA's Johnson Space Center along with the University of Texas at Tyler employed a software verification tool, EggPlant Functional by TestPlant. It is an image based test automation tool that allows users to create scripts to verify the functionality within a program. A set of edge key framework and Common EggPlant Functions were developed to enable creation of scripts in an efficient fashion. This framework standardized the way to code and to simulate user inputs in the verification process. Moreover, the Common EggPlant Functions can be used repeatedly in verification of different displays.**

***Index Terms*— Software Engineering, Verification, Computer Programming, Image Recognition**

I. Introduction

One of NASA's current objectives is advancing the state of spaceflight to Mars. For several years, NASA and its collaborators have sent orbiters, landers and rovers to learn more about the planet to lead the way for possible human exploration and habitation [1]. The knowledge gathered from the data collected so far has shown that a manned mission to Mars is feasible. Thus, NASA's Orion spacecraft is being designed and built to take humans farther into deep space. Orion is planned to be the exploration vehicle to take a human crew to space as well as to provide emergency abort capability, sustain the crew during the travel, and ensure safe re-entry from deep space due to extremely high return velocities [1, 2]. Although mission control will be monitoring the spaceflight and providing instructions to the crew, they will need a way to monitor the vehicle status and to control the vehicle independently, especially in emergency situations including loss of communication. The latest spacecraft is equipped with three main digital cockpit displays, shown in Figure 1, that serve as the main interfacing component to monitor and control [3, 4]. In order to ensure that the software operates reliably and consistently, rigorous regression testing must be carried out. NASA's Rapid Prototyping Lab (RPL) at the Johnson Space Center (JSC) is at the forefront of building and testing the flight software for the cockpit displays [5].

The RPL has a generic display format software engine to render a mock-up of the Orion cockpit displays from external files. An example of the display software is presented in Figure 2 [5]. These files describe the selected features, positions, and properties of all on-screen elements. The display formats can be created or modified without changing or recompiling the software engine. The repetitive use of display functionality on different displays does not require the code to be repeated in each display format, but rather reuse of the code in the engine can be specified with a few symbols in the external definition file [6]. The RPL, in collaboration with the University of Texas at Tyler, has been testing the different functionalities of this cockpit display software using a selected software verification tool called Eggplant Functional.

II. Software Verification Tool

EggPlant Functional was chosen by the RPL as one of its software verification tools to test their cockpit display simulation software [7]. The advantage of EggPlant Functional is that it can be automated to run tests of simulated user interactions on the system under test (SUT) without actual Human/Physical actions. To do this, EggPlant Functional connects remotely to the SUT and runs programs known as scripts which are a series of commands within a file that is capable of being executed without being compiled. These scripts are written in SenseTalk, a proprietary language built by TestPlant [8]. The scripts emulate user interactions on the system such as mouse clicks and keyboard entries. Advanced image recognition technology in EggPlant Functional allows testers the ability to create decision structures that can mimic complex user behaviors. Using the built-in image recognition functions, EggPlant can be scripted to react as the user would to certain events in the software, or to verify if the SUT is acting in accordance with expected results. Test results can be logged into external log file outputs and images of the SUT can be captured during the testing process. Applications are tested from the user's point of view, which provides for a higher fidelity test.

The proprietary scripting language SenseTalk is designed to mimic natural human language, which simplifies the writing of scripts. SenseTalk follows the structure for high-level languages that are closer to human languages and further from machine languages. Listing 1 is a simple script to test to see whether the NASA Logo can be found on the NASA website.

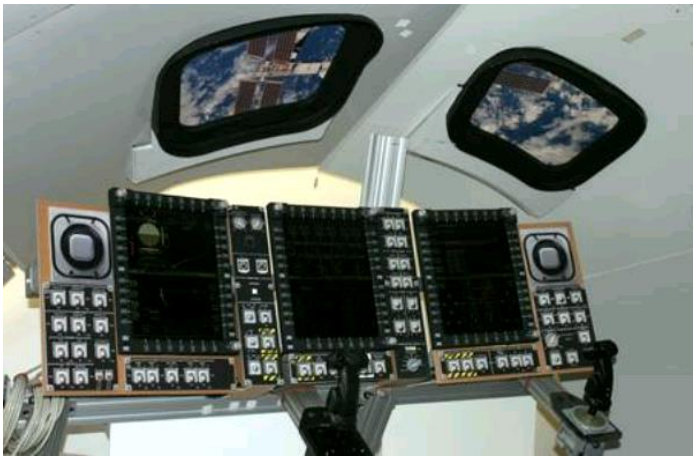

Figure 1. Mock-up of glass cockpit displays [3].

Figure 2. Example of the three Orion cockpit displays [5].

```
Click(Image:"Internet_Browser_Icon", waitfor:5)
Click(Image:"Internet_Address_Bar", waitfor:5)
TypeText "www.nasa.gov"
Wait(5) // Wait for website to load
If ImageFound(Image:"NASA_Logo", waitfor:5) then
LogSuccess "Found NASA Logo!"
Else
LogError "NASA Logo not found."
End if
```

First, the script will search the desktop of the SUT for the internet browser icon image, and click on that image to launch the browser. Next, the script will search for the image of the address bar and make the address bar active with a mouse click. Then, using the TypeText command to send keyboard entries to the SUT, the script will type out the NASA website address into the address bar. The script will wait 5 seconds for the website to load. Depending on whether or not the NASA logo can be found by EggPlant Functional, the script will report either a success or a failure in its log output file.

EggPlant Functional can be automated to perform several iterations of testing and verification. The resultant reduction in time due to scripted automation is critical to the RPL's quick turnaround time. The iterative testing also allows the RPL to stress-test its software systems to ensure functionality in the event of a change or update to the software code [9].

In addition to the SenseTalk scripting in EggPlant, the Python programming language was utilized in order to automate the writing of SenseTalk scripts. Python has powerful file creation and text parsing capabilities that were extensively used in these testing processes. By creating Python scripts, the team was able to parse configuration files for pertinent data and automate the creation of SenseTalk scripts as text files and the directories for these scripts, which are known as EggPlant suites. Since many SenseTalk scripts were repetitive in nature and required specific data strings from text-heavy files, coding these scripts by hand would have been very time-intensive. Automating the creation of these EggPlant suites and scripts represented yet another means of reducing time and increasing efficiency.

III. Methodology

The RPL has developed a software simulation program of the Orion cockpit displays known as "RPL Sim". It can simulate the function of the cockpit displays completely in software. This program can be loaded with insert files that determine the display unit's telemetry, so the software can simulate various scenarios during the Orion missions such as ascent and descent. RPL Sim can also launch a Graphical user interface (GUI) called RPL Graphics, the visual representation of the actual display unit hardware. The display units can be interacted with by either entering keystrokes or clicking on-screen elements of RPL Graphics.

RPL Graphics simulates the functions of the actual cockpit hardware in its entirety, which includes the various displays available to the operators. The displays can show relevant information to the crew, such as data pertaining to vehicle systems, propulsion systems, and electrical systems. One navigates between these displays by using the edge keys, which are the grey rectangular buttons in Figure 3, surrounding the Display Unit (DU). These displays can also be used to change settings to the aforementioned systems and to also display electronic procedures (eProc). The electronic procedure system "assists crew members by highlighting vehicle states on a display and cueing up appropriate displays, pop-ups, and commands" [10]. In addition, eProc is "linked to fault messages, so that crew members can quickly access procedures to any message that appears on the fault summary or log displays" [10]. Thus, eProc allows the crew to rapidly assess an emergency scenario and enact the proper steps to rectify the situation. The advantage of eProc is that it supplants hardcopy manuals, resulting in a reduction of launch weight by the order of hundreds of pounds as no printed material needs to be brought onboard. A crewmember can access the eProc display to pull up an onscreen operations manual that can walk the user through the various stages of the mission.

There are three main forms of crew interaction with the display: a cursor control device (CCD), a cursor knob (also known as a "twizzle"), or edge key presses that traverse through selectable elements on the screen. Pressing edge keys is the most direct method of control, and can be performed in RPL Graphics either by clicking the image of the edge key or by entering in a pre-programmed keyboard combination. The CCD is to the pilot's left and complements the joystick. On the CCD are several buttons, a bidirectional rocker switch, and "a four-way 'caged' castle switch on the CCD…designed to travel only to controllable elements toward decreasing erroneous

cursor movements" [11]. The switches allow the user to transfer between the various displays and popups.

Popups are the menus that appear onscreen when certain selections are made within the displays. For example, when the user is in the Main Propulsion System (MPS) display, a popup appears when the edge key is selected for "Engine Shutdown". This popup has one enumeration, or choice, which allows the user to shutdown a given engine. There are seven main classes of popups depending on the enumerations or output types. When one of the popup types is selected, a text string known as an MSID will be sent to the RPL Sim window. When an enumeration is selected by the user, a number value known as a CMD Value is sent to the RPL Sim window. By checking to see if these MSID and CMD values match up with the default value, we can test to see if the popups and enumerations are working properly.

The main difference between the Display Popup script and a Primary Flight Display (PFD) script is that the PFD script is iterative. For example, the PFD EggPlant script involves incrementing a number on the display, such as a degree of rotation of the spacecraft's position, and verifying the display is working nominally. The Display Popup script involves automating the same edge key presses or cursor knob/twizzle twists needed to produce a specific display configuration. These commands are hardly iterative, and therefore the code automating the edge key strokes or twizzle twists must be hard coded outright. In simple terms, the script automates these simulated user inputs to obtain a specific popup in order to verify its functionality.

IV. Results and Discussion

Edge key usage was automated for the three display units. The three display units are DU1, DU2 and DU3. This automation assisted in the testing of the RPL display unit simulator. The edge keys were invoked using assigned edge key Keystroke identifiers, as shown in Figure 3. This allowed edge keys to be used without actually physically invoking them, and allowed for reusable code that does not rely on edge key image location recognition. A hard coded framework was written to save time from rewriting the same code over and over. This framework can then be called into any new EggPlant script via keyword for Sensetalk "put" and assigning it a path and variable name. This script has the ability to enumerate through displays using edge keys via an assigned keystroke identifier per edge key, with no physical actuating of edge keys needed.

Each display unit was broken down into sections as seen below. The six main sections are upper left, upper right, lower left, lower right, top navigation, and bottom navigation. Figure 4 shows also miscellaneous navigation buttons that are included in the framework. These include enter, cancel, and twizzle.

The framework scripts contain variables that represent different keystroke combinations, each of which correspond to an onscreen function like actuating an edge key or turning the twizzle knob left.

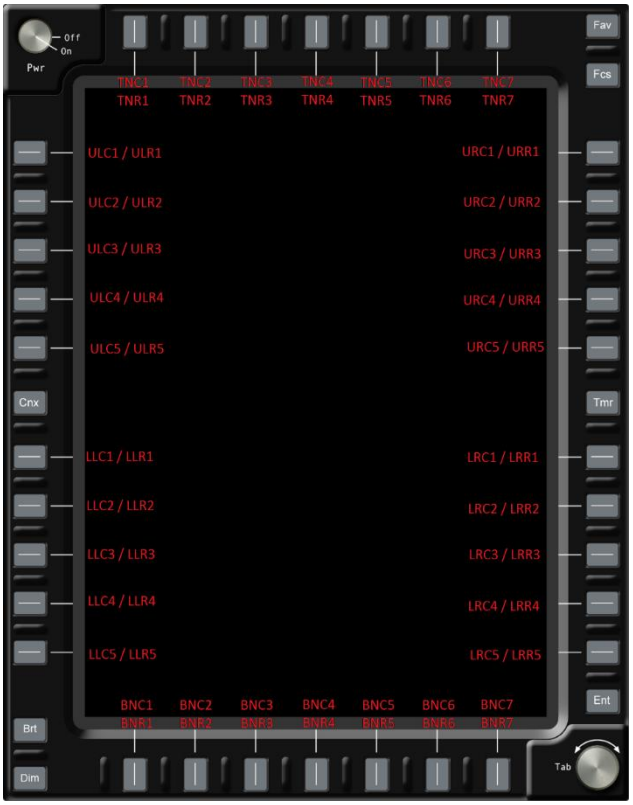

Figure 3. Edge keys of a display unit template

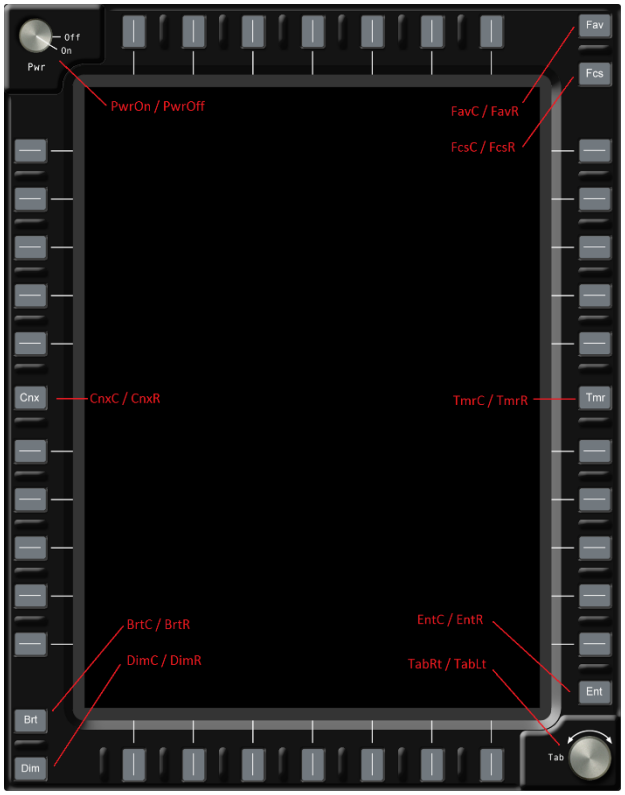

Figure 4. Navigation buttons of a display unit template

Figure 5 is a chart of the upper left section and the edge key variable names. There is a separate variable for each edge

key's click and release command as illustrated by each DU column.

Prebuilt functions were also added to navigate to specific displays quickly. Figure 6 is the list of prebuilt functions that have been made.

To be able to use the framework in a script, it must be called into the script at the beginning of the script. Then, it is possible to start calling on it to press edge keys. Listing 2 is a sample script that calls in the framework, navigates to the MPS Display, and clicks and releases two edge key buttons.

Desired EdgeKey Function	Variable Name	DU1	DU2	DU3
EK_Upper left click 1	ULC1	*30	,30	.30
EK_Upper left click 2	ULC2	*31	,31	.31
EK_Upper left click 3	ULC3	*32	,32	.32
EK_Upper left click 4	ULC4	*33	,33	.33
EK_Upper left click 5	ULC5	*34	,34	.34
EK_Upper left release 1	ULR1	'30	;30	\30
EK_Upper left release 2	ULR2	'31	;31	\31
EK_Upper left release 3	ULR3	'32	;32	\32
EK_Upper left release 4	ULR4	'33	;33	\33
EK_Upper left release 5	ULR5	'34	;34	\34

Figure 5. Assigned edge key variable names

Display Screen	Variable Name
EPS Battery Display	EPS_Battery
EPS Heater Display	EPS_Heater
EPS 28 Volt Display	EPS_28Volt
Flight PFD Display	Flt_PFD
Flight MPS Display	Flt_MPS
Flight Landing Display	Flt_Landing
Veh Rad Display	Veh_Rad
Veh SM Sep Display	Veh_SM_Sep
CW Log Display	CW_Log
Prop CM RCS Display	Prop_CM_RCS
Img Images Display	Img_Images
GNC Switch Display	GNC_Switch

Figure 6. List of Prebuilt functions

```
put "C:\Users\user\Desktop\DisplayUnit_KeyStroke
_Framework.suite\Scripts\DU1" into DU1

//Navigates to MPS Display
put DU1.Flt_MPS

//Click and release 2 edge key buttons
put DU1.URCR1
put DU2.LRCR3
```

After the framework for the DU edge keys and display calls was written and verified, it was time to write the scripts to test the MPS Display. The MPS Display has four sections that correspond to the four engines of the propulsion system that are active during ascent. Each of the four engines has two options, Engine Shutdown ("Eng S/D") and Redline Enable/Inhibit ("Redline Ena/Inh"). Eng S/D is a "NORMAL" type popup that sends only one enumeration to shut down the engine. Redline Ena/Inh is a "SEND SINGLE" type popup with a total of two enumerations. The user can either Enable or Inhibit the Redline warning of a given engine.

A script called "Common EggPlant Functions" was written that acted much like our edge key framework. Common Eggplant Functions contained several methods that were used repeatedly by our verification scripts. There were methods for selecting the popup, sending the right enumeration, and capturing the MSID and CMD value after the enumeration was sent. These methods made up the backbone of our MPS Display verification script.

Our MPS Display test script worked as follows. The Edge Key Framework and Common EggPlant Functions were called at the beginning of the script. A display call was then made for the MPS Display. The script iterated through all the popups and enumerations of the MPS Display for each engine in numerical order. The MSIDs and CMD values were captured, output to a text file, and compared to the expected values. Afterwards, the user can access the log files and determine if the test was a success or a failure, depending on whether the captured values were the same as the expected values.

The MPS Display test script was set to run numerous times by EggPlant Functional. Successive testing proved that the scripts were working and were able to obtain the corresponding values consistently

V. Conclusion and Future Work

EggPlant Functional and SenseTalk proved to be highly useful tools for automating the testing of the Orion cockpit display simulators. The Edge Key framework and Common EggPlant Function scripts also enabled us to create scripts in a more efficient manner. One of the main goals in the testing process was to write modular, re-usable code that can be applied to the different displays, popup types, and enumerations that one encounters. The framework standardizes the way in which we code and simulate user inputs, and the Common Eggplant Functions can be used repeatedly in tests of other displays.

Moving forward, we hope to use the tools that we have created to test the other displays and popup types in RPL Sim. We also want to run repetitive tests to prove that our scripts are functional and obtaining the correct results. Moreover, this iterative testing will validate reliability and consistency of the Orion cockpit display simulation software performance. The final step will involve a formal verification process to be performed independently as part of an overall quality assurance process before the Orion spacecraft takes flight.

ACKNOWLEDGMENT

We would like to thank University of Texas at Tyler Office of Research and Technology Transfer. We would also like to acknowledge the efforts of Steven Auzenne, Jonathan Vidana, Jeffrey Fox and Patrick Henry within the RPL.

REFERENCES

[1] NASA, "Journey to Mars," 30 September 2016. [Online]. Available: https://www.nasa.gov/topics/journeytomars/index.html. [Accessed 26 April 2017].

[2] K. L. Holden, J. L. Boyer, N. Ezer, K. Holubec, A. Sándor and J.-P. Stephens, "Human Factors in space vehicle design," Acta Astronautica, vol. 92, no. 1, p. 110–118, 2013.

[3] C. Murray, "Inside NASA's glass Orion cockpit," 3 August 2015. [Online]. Available: https://spaceflightuk.com/2015/08/03/inside-nasas-glass-orion-cockpit/. [Accessed 29 April 2017].

[4] NASA, "Orion Simulations Help Engineers Evaluate Mission Operations for Crew," 22 February 2016. [Online]. Available: https://www.nasa.gov/feature/orion-simulations-help-engineers-evaluate-mission-operations-for-crew. [Accessed 29 April 2017]..

[5] NASA, "Orion: Cockpit," 1 December 2014. [Online]. Available: https://archive.org/details/Orion_Cockpit. [Accessed 26 April 2017].

[6] NASA JSC, "Rapid Prototyping Lab (RPL) Generic Display Engine," 1 October 2014. [Online]. Available: http://www.techbriefs.com/component/content/article/ntb/tech-briefs/information-sciences/20738. [Accessed 26 April 2017].

[7] Testplant, "eggPlant Functional," June 2013. [Online]. Available: https://www.testplant.com/eggplant/testing-tools/eggplant-developer/. [Accessed 29 April 2017].

[8] Testplant, "About SenseTalk," [Online]. Available: http://docs.testplant.com/?q=about-sensetalk. [Accessed 29 April 2017].

[9] S. Auzenne, M. Dumantay, J. Vidana, M. Issa, M. A. R. Biswas, M. Souris and L. Morin, "Automated Testing of Orion Cockpit Displays using EggPlant Functional and Python Programming," in ASEE Gulf-Southwest Section Anuual Conference, Dallas, 2017.

[10] NASA, "eProc Electronic Procedure System for Spacecraft Glass Cockpits (eProc System)," [Online]. Available: https://software.nasa.gov/software/MSC-25186-1. [Accessed 26 April 2017].

[11] M. C. Dorneich, J. A. Lancaster, C. J. Hamblin, O. Olofinboba and R. E. Demers, "Deriving Cursor Control Device Expectations for the Orion Crew Exploration Vehicle," in Human Factors and Ergonomics Society Annual Meeting, San Francisco, 2010.

[12] M. Dumantay, S. Auzenne, J. Vidana, M. Issa, M. A. R. Biswas, M. Souris and L. Morin, "Verification of Orion's Cockpit Displays Using EggPlant and Python," in IEEE DUAL CONFERENCE Of INNOVATION and AUTOMATION, Houston, 2016.