Develop a Model Component

Tyler Ensey

Kennedy Space Center

Major: B.S. in Electrical Engineering

KSC FO Fall Session

Date: 29 07 2013

Develop a Model Component

Tyler S. Ensey[1]
University of Central Florida, Orlando, FL, 32828

During my internship at NASA, I was a model developer for Ground Support Equipment (GSE). The purpose of a model developer is to develop and unit test model component libraries (fluid, electrical, gas, etc.). The models are designed to simulate software for GSE (Ground Special Power, Crew Access Arm, Cryo, Fire and Leak Detection System, Environmental Control System (ECS), etc...) before they are implemented into hardware. These models support verifying local control and remote software for End-Item Software Under Test (SUT). The model simulates the physical behavior (function, state, limits and I/O) of each end-item and it's dependencies as defined in the Subsystem Interface Table, Software Requirements & Design Specification (SRDS), Ground Integrated Schematic (GIS), and System Mechanical Schematic (SMS). The software of each specific model component is simulated through MATLAB's Simulink program. The intensive model development life cycle is as follows: Identify source documents; identify model scope; update schedule; preliminary design review; develop model requirements; update model scope; update schedule; detailed design review; create/modify library component; implement library components reference; implement subsystem components; develop a test script; run the test script; develop users guide; send model out for peer review; the model is sent out for verification/validation; if there is empirical data, a validation data package is generated; if there is not empirical data, a verification package is generated; the test results are then reviewed; and finally, the user requests accreditation, and a statement of accreditation is prepared. Once each component model is reviewed and approved, they are intertwined together into one integrated model. This integrated model is then tested itself, through a test script and autotest, so that it can be concluded that all models work conjointly, for a single purpose. The component I was assigned, specifically, was a fluid component, a discrete pressure switch. The switch takes a fluid pressure input, and if the pressure is greater than a designated cutoff pressure, the switch would stop fluid flow.

Nomenclature

GIS	=	Ground Integrated Schematic
COTS	=	Commercial off the Shelf
GSE	=	Ground Support Equipment
LCS	=	Launch Control Systems
SMS	=	System Mechanical Schematic
MTR	=	Model Test Report
MRDD	=	Model Requirements and Design Document
GUI	=	Graphical User Interface
UCTS	=	Universal Coolant Transporter System
SLS	=	Space Launch System
PRSD	=	Power Reactant Supply and Distribution
DPI	=	Default Physical Interface
P	=	pressure
V	=	flow velocity
T	=	temperature
k	=	k-value
q	=	q-value
z	=	elevation
A	=	area

[1] KSC FO Intern, NE-C1, Kennedy Space Center, University of Central Florida.

ρ = density of fluid
h_L = Head Loss
g = gravity (32.2 ft/sec^2)
subscript 1 = input of container
subscript 2 = output of container

I. Introduction

THE development of models is crucial for testing software before it is implemented into the hardware. Since the software cannot be tested on the hardware immediately, because of possible issues that could arise, such as, in extreme cases, the possibility of breaking the hardware in question or other hardware as a result of the effects the software has on the hardware. Instead, a model of the hardware is made in an environment, so that the effectiveness of the software can be tested in a safe and feasible environment.

Each model is first made independently and tested, before it is incorporated into either another model, or into the larger scale model, known as the Ground Integrated Schematic (GIS). This takes much cooperation and coordination, something I learned a great deal about during my internship. This is because sometimes a different modeler will create the system test for a model that was developed by a separate modeler. Therefore, the modeler who created the model needs to put in adequate requirements, to make it easier for the system test to be made. Also, the modeler designing the system test needs to be sure to coordinate with the original modeler, to be sure that any changes made to the original model will not change the desired outcome of the component. Another consideration that needs to be taken into account is the effect on the schematic. The modeler needs to be certain of the requirements of their model and what should and should not be included as the inputs, outputs, subsystems, parameters, etc.

II. Description

On my first day at NASA Kennedy Space Center, I was informed that I would be working towards designing and testing my own fluid component, and also performing system tests on models that were already developed. The fluid component that was assigned to me was a discrete pressure switch. The first requirement was to do adequate research on the discrete pressure switch, the environments I would be working with, and all other considerations that go into being a model developer, in order to thoroughly understand the operations I would be performing and understand why a pressure switch would be necessary in a system.

A. Simulink

Every component is modeled in the Simulink environment. Simulink is a Commercial off the Shelf (COTS) simulation tool, created by the company MathWorks. Simulink is used for multi-domain simulation and developing model based designs. Simulink provides the user with an environment that is both one that provides the user with both a visual and interactive experience. Customizable block libraries are another feature that Simulink graces to the modeler. These libraries are used to design, simulate, and test multiple time-varying systems within the Simulink environment. These systems include communications, controls, signal processing, video processing, and image processing.[1] Many different industries utilize Simulink for the development of mathematical models of physical dynamic systems. In specific, my coworkers and I, used/are using Simulink and multiple libraries, libraries provided within Simulink by MathWorks and also a set of KSC developed custom libraries, to construct models of Ground Support Equipment (GSE).[2]

There many ways of doing mathematical calculations. In this case, there are five recognizable ways designated by the Launch Control Systems (LCS) simulation group: Simulink blocks, embedded MATLAB functions, stateflow diagrams, Simulink look-up tables, and truth tables. There are certain requirements for each of these. Simulink blocks should contain equations that are broken up into logical, contained units, and the equation should be presented in a readable way. This is often achieved using the Simulink "Subsystem" block. Simulink "Integrator" blocks are typically used within the developing community to represent physical quantities, such as pressure, temperature, etc. Also, the parameters "Upper Saturation Limit" and "Lower Saturation Limit," of the "Integrator" block, should be populated with acceptable limits of the physical quantity. For example, most negative quantities would not be expected in a real world situation, so most "Integrator" blocks would have a "Lower Saturation Limit" of "0." Lookup tables can be helpful, but the developer should not import data into lookup tables using Excel files. Instead, the file should be converted into a .csv file and then "csvread" should be used in the callback section of the model to import data. Truth tables are useful as decoders and controllers, especially when used in multiple places.[3]

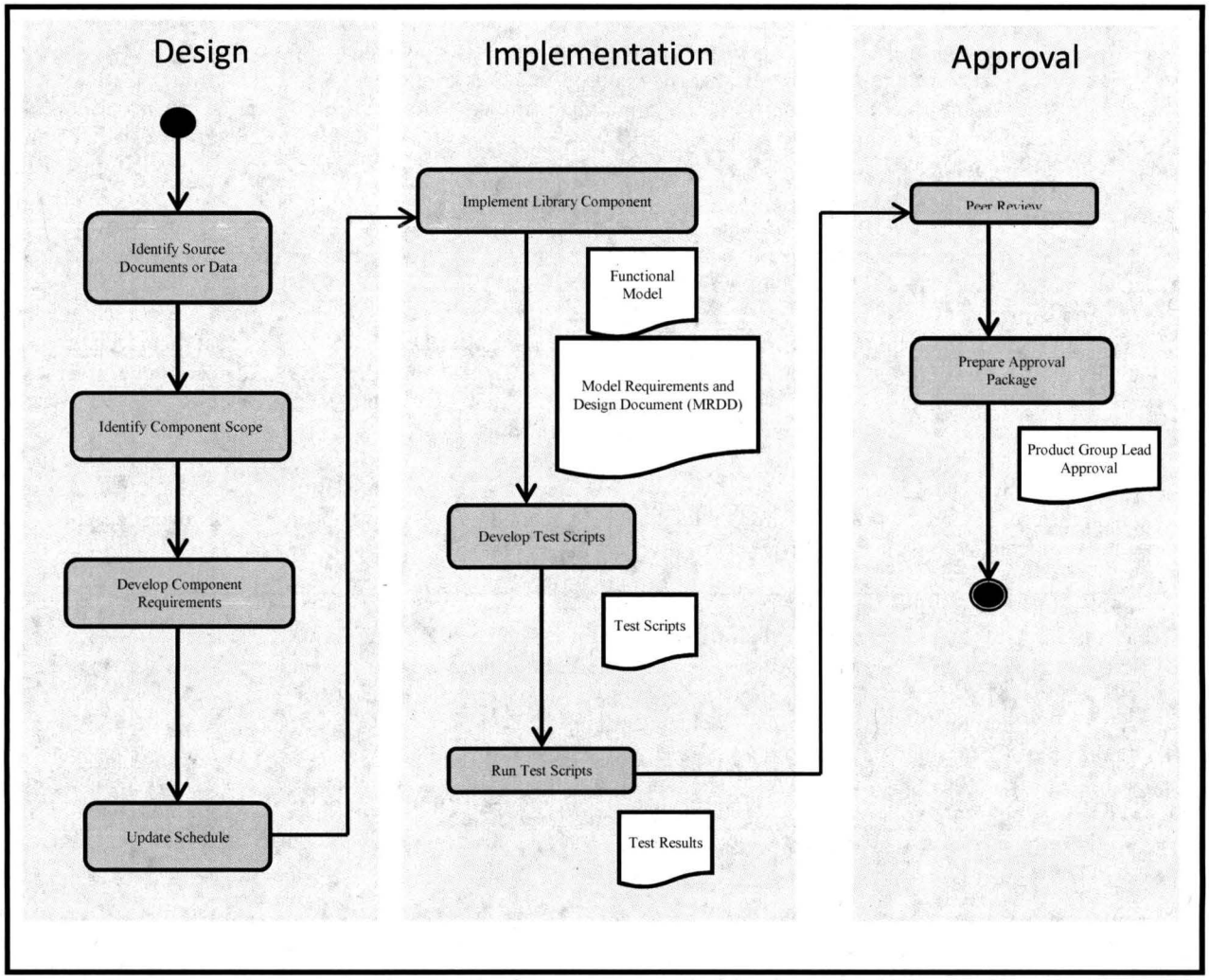


Figure 1. GSE Library Component Lifecycle. *This figure represents all of the planning and execution during the lifecycle of a GSE Library Component.*

B. GSE Library Component Lifecycle

Model Development for a GSE model is broken into three phases: Design, Implementation, and Approval. A visual representation of this process is shown in Fig. 1.

The Design phase is focused on establishing structure and external interfaces. These models represent components of the GSE Subsystem and they, without exception, must coincide with the LCS Simulation Modeling Standard. Design is begun once the necessity for a GSE component is recognized, and a schedule for the development has been created. Source documents must first be identified in the Design phase. These documents may include: GIS/System Mechanical Schematic (SMS), which identifies the manufacturer and model of a component and its functional parameters; Design Dictionary, which includes cutsheets of the specific component type and function parameters; and/or, ConOps, which describes the function of the component. Component scope is then identified, which includes layout and unique feature identification. For layout, the user must identify internal structures the will help in easily showing the complexity of certain interactions. Some great things to consider are grouping blocks, within Subsystems, based on their function, and also the reusability of the component, as simplicity is the key if it will be used in multiple other components. For unique feature identification, the user must identify what internal functions are unique, meaning the component cannot be developed by using readily available library blocks or components. Requirements must be developed within the Design phase. There are three types of requirements that should be concluded before moving to next phase: Test Scenario Requirements, which establishes points of verification, to ensure that the component being testes functions as desired; Training Scenario Requirements, which establishes fail-safe requirements; and, finally, Functional Requirements, which are the

requirements that the component must meet in order to be of use to the GSE Subsystem Model. The schedule is then updated, as needed, and the modeler moves on to the next stage, known as Implementation.

The Implementation stage is focused on developing the component and the test scripts. Common-use components, subsystem-specific components, and Simulink Primitive blocks should be arranged into Simulink approximations. Test Scripts must then be developed. Draft steps are created to verify the software is behaving accurately. Draft steps must also be produced to test the interface function, given that there are explicit interfaces with another block. The Test Script is then run, and the test data is recorded in a Model Test Report (MTR). A Model Advisor check will verify that Simulation Standards are met. Finally, a Model Requirements and Design Document (MRDD) will show the requirements, structure, parameters, and referenced libraries within a single document, and the user may now move on to the Approval phase.

The Approval phase is focused on reviewing the GSE Component model and approving it for use within the schematic. In the Approval phase, peer review will commence. The MRDD, MTR, and Test reports are distributed to the Simulation team, through a Graphical User Interface (GUI), known as AccuRev, so that peer review can begin. Comments are gathered pertaining to the validity of the requirements, implementing logic, and test results. Once a response from two SIM Team members and a Subject Matter Expert, appropriate changes are made to the component, and the MRDD, MTR, and Model Advisor Report are regenerated. The final step is to prepare and submit the Approval Package. This package is objective evidence of the successful completion of the Library Component Peer Review. Once the SIM Product Group Lead approves the model, the development of that component is officially complete, and can now be used in the GSE Subsystem model.[4]

C. Universal Coolant Transporter System (UCTS)

The specific GSE that the pressure switch will be used for is known as the Universal Coolant Transporter System (UCTS), which was a Space Shuttle Orbiter support system, but is being upgraded and repurposed for the Space Launch System (SLS) rocket. The UCTS is used for two main services, coolant servicing and Power Reactant Supply and Distribution (PRSD) servicing. Coolant servicing is used once the Orbiter lands; the Ammonia Boiler, which cools the avionics and payloads on board the orbiter, is shut off upon landing, so the Orbiter is connected to the UCTS in order to keep the avionics and payloads from getting damaged by overheating. PRSD servicing allows for the deservicing of propellants.[5] The operating system of the UCTS is shown in Fig. 2. The four main parts of the system, the primary loop, the secondary loop, the refrigeration subsystem, and the pump/reservoir operation are described below:

PRIMARY LOOP

R-124 in the primary loop can exit the Orbiter at 30 to 100 degrees F with a maximum heat load of 128,000 Btu/h. Experience in operations has shown that the Orbiter return temperature is maintained at 60 ^{O}F. The R-124 return fluid first encounters the systems' primary circulation pump, located on the reservoir / pump subsystem. After which the flow is split into two paths. A portion of the primary loop fluid is routed to the temperature controlled mixing valves, located on the circulation control subsystem. This path is the source of the warm fluid that is mixed with the cold fluid which is routed to the secondary loop heat exchangers where it is conditioned to 10 ^{O}F. From the heat exchangers the fluid proceeds direct to the temperature controlled mixing valves. At the

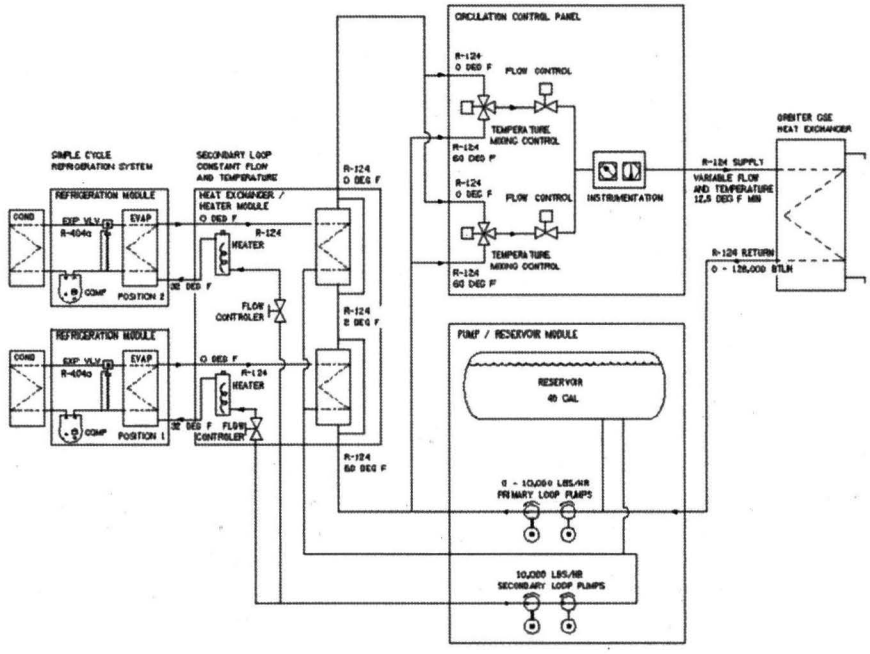


Figure 2. The Universal Coolant Transporter System. *This is the model for the operating system of the UCTS.*

temperature controlled mixing valves all or none of the fluid can be directed to flow through the secondary loop heat exchangers.

SECONDARY LOOP

The secondary loop is simple. The PLC monitors only one variable (temperature) and adjusts it through a single device, the heater, into a constant. Flow in the secondary loop is preset and fixed. Empirical data taken at OPF-3 has proven the criticality of flow rate in order to make the system work. The secondary loop eliminates all variables ensuring the greatest reliability and ease in troubleshooting. (…) As a result of the temperature controlled mixing valves, all or part of the primary loop flow can be directed to the secondary loop heat exchangers. The portion of primary loop return fluid directed to the secondary heat exchangers, flow varies, causing changes in the heat load and secondary fluid temperature; however, the thermostatically controlled heaters in the secondary loop maintain the fluid going to the refrigeration subsystems at a constant design temperature.

REFRIGERATION SUBSYSTEM

The UCTS has two identical refrigeration subsystems, piped in series into the primary loop, via the secondary loops. Each refrigeration subsystem is sized to handle 87,000 Btu/h at 110 degrees F desert conditions. Experience has shown that the maximum Orbiter heat load generated during post landing operations is 60,000 Btu/h, and averages 40,000 Btu/h. The refrigeration subsystem is basic and consists of a positive displacement discus compressor; refrigerant receiver, air-cooled condenser; suction heat exchanger / accumulator; an expansion valve and chiller barrel.

PUMP / RESERVOIR OPERATION

The primary pumps are configured in series. One pump is required for normal operation, and the second is provided for back up. Series configuration was chosen to meet any future pressure increase requirement. The pump operation is locked out by software preventing undesired operation of two pumps simultaneously. The secondary pumps are also configured in series to accommodate space allocation. (…) The reservoir maintains both the primary and secondary loops. This decision was made to accommodate space allocations. The reservoir was piped as an accumulator (out of the flow path) vs. in line to allow for system temperature stabilization of the secondary loops.[5]

D. The Discrete Pressure Switch

A pressure switch reads the input pressure of a certain fluid, allowing for normal operation. Once the pressure reaches a certain designated "cutoff" pressure, the switch will turn to open or closed, depending on the normal operation. The pressure switch is a simple but undeniably important part of the system. The pressure switch makes sure that a safe pressure is maintained, and once that safe pressure is exceeded, the pressure switch switches to the non-normal state. The fluid component reads the input pressure and delivers a value, whether Boolean(0) or Boolean (1), depending on the normal condition, to an electrical model that will calculate the necessary electrical outputs, and cutoff the flow of electrical current.

III. Fluid Component Model

A. The Library Component

Once I had done adequate research on what I was modeling, what needed to be calculated, what variables needed to be established, and how I would go about developing this model, I was required to create a library component for the discrete pressure switch. I designed a picture that coordinated with the desired image, specified in the GSE Standard document.[6] This image and my coordinating drawing is shown in Fig. 3. There is a previously developed library component known as a Default Physical Interface (DPI), which represents all of the important variables that should be inputs and calculated outputs. For fluids, these variables are pressure, P, flow velocity, V, temperature, T, k value, and q value. A DPI allows for all of the changes on these variables to be passed through a single port, which is important for the detailed schematic diagram. A DPI was connected to a subsystem, named 'DoCalc', which models the function of the pressure switch, and returns the output values for each variable, shown in Fig. 4. This is shown within the There is also a 'goto' block, which takes either a Boolean(0) or Boolean(1) value, depending on the cutoff pressure and if the input pressure exceeds that cutoff pressure, and sends it to an electrical component which will cutoff the electrical current. Temperature, k, and q are not affected by the pressure switch. Output velocity and

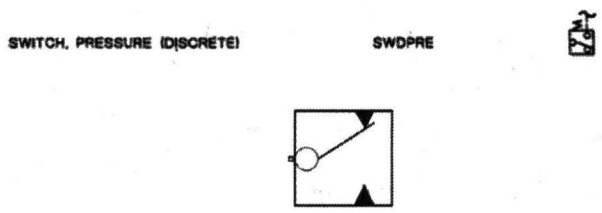


Figure 3. Mask Image. *The above figures represent the desired and actual images for the Discrete Pressure Switch. The figure on top is the figure designated in the SMS, and the figure below it is my created drawing.*

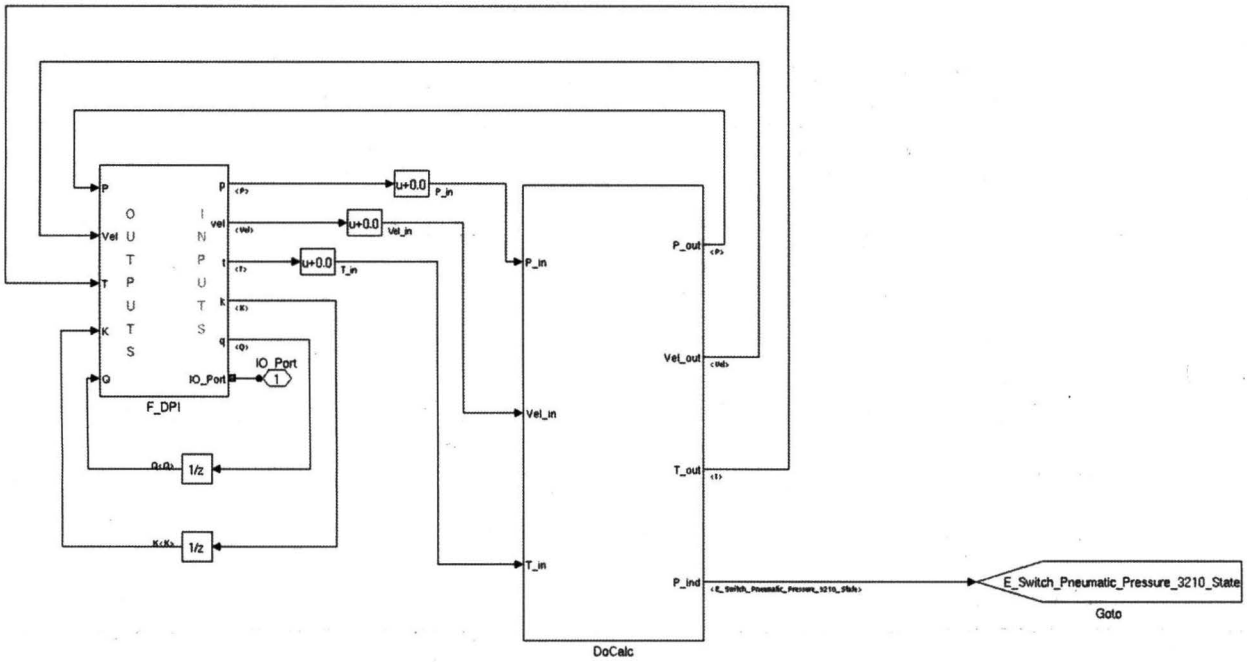


Figure 4. Model of the Connected DPI and 'DoCalc' Subsystems. *This figure represents the Discrete Pressure Switch in its simplest form (without the actual calculations, which are represented within the 'DoCalc' subsystem.*

output pressure however will differ from their relative inputs and outputs, as a result of the component. Thus, velocity and pressure must be calculated within the model. The equation for doing so is known as Bernoulli's equation. This equation is as follows:

$$P_2 = \frac{\rho_2 g}{144}\left[(z_1 - z_2) + \frac{144 * P_1}{\rho_1 g} + \frac{V_1^2 - V_2^2}{2g} - h_L\right] \quad (1)$$

Since we do not know V_2, we must calculate the output velocity first, using the following equation:

$$V_2 = \frac{A_1 * V_1}{A_2} \quad (2)$$

Equations (2) and (1) are represented within two separate subsystems within the 'DoCalc' subsystem shown in Fig. 4. Since V_2 is needed for the calculation of P_2, the output for the subsystem representing equation (2) is connected as an input for the subsystem representing equation (1). P_{ind} is also an output of the subsystem, and is connected to a 'goto' block; this output represents the Boolean(0) or Boolean(1) value that tells the electrical component what it should perform. A test model was then created with another DPI connecting the I/O port to the pressure switch's mask, and a 'from' block going to a separate output, which represents P_{ind} in the test. For more information on the complexity of the Bernoulli's equation, see Ref. 7.

B. The Test

A MATLAB script is written within the test harness, in order to test if the desired outputs are indeed being calculated. The test is then run, and the expected output values are compared to the actual output values, and a pass or fail prompt is returned to the user. A portion of the written test script is shown in Fig. 5. Once the test script passes on every iteration (the number of iterations is based on the number of user designated expressions for the length of each input, and the program determines the calculated number of possible iterations), an autotest is run. The autotest will run Model Advisor and the system test, and then produce the MTR, MRDD, System Test, and Model Advisor documents. These documents (only small portions, per their large sizes) are represented in Fig. 6. After these documents have been produced, the library component, the test model, the test script, and the documents are uploaded to AccuRev, and when ready, will be peer reviewed for acceptation.

```
for i = 1:numSteps

    VelOexp(i) = ((r1)^2 * h1 * VelIn) / ((r2)^2 * h2);
    POexp(i) = (ro * 32.2 / 144) * ((z1 - z2) + (144 * PIn / (ro * 32.2)) + (((VelIn)^2 - (VelOexp(i))^2) / (2 * 32.2)) - hL);
```

Figure 5. MATLAB Test Script. *This is a representation of a small portion of the Test Script that is used to test the functionality of the Discrete Pressure Switch.*

a)

Report name: Model Advisor

Simulink version: 7.4 **Model version:** 1.3990
System: F_Pressure_Switch_Test **Current run:** 16-Jul-2013 16:25:42

67 of 67 Passed

Check model, local libraries, and referenced models for known upgrade issues

Passed

Identify unconnected lines, input ports, and output ports

Passed

a) The Model Advisor Document

b)

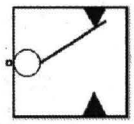
Library Component: F_Pressure_Switch

Model Requirements and Design Document (MRDD)

Date: 27-Jun-2013

Chapter 1. Root System

Diagram



F_Pressure_Switch_3210

Mask Parameters

Mask Parameter	Initial Value
Callout	3210
Max Pressure	500
Input Elevation	5
Output Elevation	5
Density	62.4
Length	10
Input Radius	4
Output Radius	4
Mu	2.25e-4
Input Length	2
Output Length	2

Requirements

<F_Pressure_Switch_R_001>

The F_Pressure_Switch_3210 block shall implement the equations for VelOut, POut, TOut, KOut, and QOut

Parameters:
r1 = input radius; h1 = input length; r2 = output radius; h2 = output length
ro = density; z1 = input elevation; z2 = output elevation

These equations are as follows:
VelOut = (r1^2 * h1 * VelIn) / (r2^2 * h2);
r1 = input radius, r2 = output radius; h1 = input length, h2 = output length;
ro = density; z1 = input elevation, z2 = output elevation; L = Length; mu = Viscosity
POut = (ro * 32.2 / 144) * ((z1 - z2) + (144 * PIn / (ro * 32.2)) + (VelIn^2 - VelOexp(i)^2) / (2 * 32.2)) - hL)
hL = Head Loss = ((f * (L / 12) * (Dia / 12)) * VelIn^2 / (2 * 32.2)
f = (64*mu)/DVp
DVp = (Dia / 12) * VelIn * ro
Dia = 2*r1
TOut = TIn
KOut = KIn
QOut = QIn

If PIn > P_max (cutoff pressure), then the block will generate a boolean integer output of 1.
Otherwise, the block will generate a boolean integer value of 0.

End <F_Pressure_Switch_R_001>

b) The MRDD Document

c)

Limit Check

Test Variable	Operator	Limit	Evaluates To
O1 blue	==	O1exp 1	TRUE
VelO 400	==	VelOexp 400	TRUE
TO 1	==	TOexp 1	TRUE
KO 1	==	KOexp 1	TRUE
QO 1	==	QOexp 1	TRUE

Test Variable	Expected Value	Tolerance Type	Tolerance Limit	Evaluates To
PO 549.96	POexp 549.95	Absolute	POexp_tol 0.54999	TRUE

Iteration 18 Passed

Post Test

F_Pressure_Switch_Test Done

Generated Files

The following files were generated in /net/users/kmoey/accurev/SIM_MODEL_libs/Model_Development/libs/fluid/test/F_Pressure_Switch/Test/

Open	Filename
Test Results	F_Pressure_Switch_Test_results.mat
Test Report	F_Pressure_Switch_Test_report/F_Pressure_Switch_Test_report.html

Final Test Status

Property	Value
Start Time	16-Jul-2013 16:28:01
Stop Time	16-Jul-2013 16:28:25
Iterations Completed	18
Iterations Passed	18
Iterations Failed	0
Final Status	Passed

c) The Test Script Document

d)

Library Component: F_Pressure_Switch

Math Model Test Report (MTR)

Date: 16-Jul-2013

Chapter 1. Report Location

Accurev Location : ./libs/fluid/test/F_Pressure_Switch/Test/F_Pressure_Switch_MTR.html

Chapter 2. General Software Environment and Assumptions

1.1 Test Environment
Testing will be performed on an individual basis at dedicated workstations, such as an office environment.
Whenever possible, testing will be performed with Mathworks Simulink/Matlab suite of programming and simulation tools/environment.

It is recognized that testing may be required that exceeds the scope of Simulink test, such as TRICK interface. In these cases, the appropriate testing tools are permitted.

1.2 Assumptions
All members of the test team are familiar with testing tools, such as the Mathworks Simulink/Matlab suite of software tools or the TRICK environment.
All members of the test team have compatible versions of the testing tools.
All members of the test team have access software same repository, such as Accurev, KDDMS, etc.
All pathname locations in this report are relative to the Accurev path :

./libs/fluid/test/F_Pressure_Switch/Test

1.3 Scope Purpose
This test report provides a summary of the tests executed against a particular component for the purpose of verification.
The Test Summary section identifies the tests that have been run, the requirements satisfied with each test, any required input files for the test, the location of the detailed test report in accurev, test execution information and a Final Status for the Pass/Fail criteria for the test.

Chapter 3. Test Summary

Location: slprj/modeladvisor/F_Pressure_Switch_Test/report.html
Final Status: passed

d) The MTR Document

Figure 6. Documents for Discrete Pressure Switch. *These are the MTR, MRDD, Test Script, and Model Advisor Documents for the component that represent all desirable aspects of the component.*

IV. Conclusion

My GCS component is currently awaiting peer review and will hopefully be reviewed and returned to me before my internship has concluded. This internship has given me great insight as to what kind of workplace career can be expected once my college career has concluded, and I can happily say that I look forward to every second of it. It's a completely different lifestyle than college, a more enjoyable lifestyle that is. Every day starts off early, and you are done in time to enjoy your afternoons. You have your work life and your home life, unlike college where work life and home life turns into one big blur of work, to a point where you do not remember which is which. That's not to say you can slack off on the clock. You are representing you and everything you stand for, something that I am unwilling to compromise. When I'm here, it's not about getting by, it's about showing everyone, including myself, the type of employee that I can be. So, if you are reading this as a future intern, money should not be your only motivation; whether paid for or not, your pride and fortitude should drive you to be the best employee and coworker you can possibly be.

Acknowledgments

I would like to thank a few people who were crucial to my success during my internship at KSC. First, I owe Cheryle Mako my greatest gratitude for giving me this life changing opportunity, and giving me a chance to succeed, while being able to see what my future has in store. Next, I would like to thank Lien Moore for all her hard work and diligence towards everything she does. Although she is a perpetually busy employee, she consistently ensured time to assess if I had any problems or questions in which she could provide solutions. She undoubtedly gave me the feeling that she genuinely was concerned with my well-being, something that I did and always will cherish deeply. Finally, I would like to thank Camiren Stewart; Cam was perhaps one of, if not the, most helpful member of NE-C1. I felt that he could relate to me in a way that no one else could, because he was fairly recently an intern at KSC, but at the same time he is intelligent far beyond his years. The fact that he was in my shoes just a few years before me, as I sit in a similar place and type on a similar computer as he did, is unbelievable. He is the perfect model of what I aspire to be in a year, when I graduate college. He is an asset that NASA should be immensely proud to have, but should be appalled at the thought of losing.

References

[1]MATLAB, Ver. 7.9.0 (R2009b), The MathWorks Inc., Natick, MA, 2009.

[2]Grant, K., "Simulation CSCI Software Design Description," Document Number: K0000111998-GEN, Archive URL: https://sp.ksc.nasa.gov/ne/simulation/Shared Documents, NASA Kennedy Space Center, Merritt Island, FL, 2013.

[3]"LCS Simulation CSCI Modeling Standard," Document Number: GOP507228, Archive URL: https://sp.ksc.nasa.gov/ne/simulation/Shared Documents, NASA Kennedy Space Center, Merritt Island, FL, 2013.

[4]Moore, L., "E2E CC SDP Volume 4," Document Number: K0000143174-PLN, Archive URL: https://sp.ksc.nasa.gov/ne/simulation/Shared Documents, NASA Kennedy Space Center, Merritt Island, FL, 2013.

[5]Katz, M., "Universal Coolant Transporter System Technical Manual & Operating Criteria," URL: https://usa2.usa-spaceops.com/fl/ucts/a80k59593_index.htm, NASA Kennedy Space Center, Merritt Island, FL, 2013.

[6]"Graphic Symbols for Drawings Part 2 Standard for Ground Support Equipment," Document Number: KSC-STD-152-2C, Archive URL: https://sp.ksc.nasa.gov/ne/simulation/Shared Documents, NASA Kennedy Space Center, Merritt Island, FL, 2013.

[7]"Bernoulli's Equation," Princeton University, URL: http://www.princeton.edu/~asmits/Bicycle_web/Bernoulli.html, 29 July 2013.