Software Development for Remote Control and Firing Room Displays

Zambrano Peña, Jessica (KSC) [KSC NASA Intern]
Kennedy Space Center
Computer Science & Mathematics

KSC FO Summer Session
07 18 2014

Software Development for Remote Control and Firing Room Displays

Jessica Zambrano Peña

University of Texas at Brownsville – Brownsville, TX 78520

Nomenclature

KSC	= Kennedy Space Center
GFAST	= Ground Flight Application Software Team
SRDS	= Software Requirements & Design Specification
TBD	= To Be Determined
CUI	= Compact Unique Identifier
GUI	= Graphical User Interface
ILOA	= Integrated Launch Operations Applications
DE	= Display Editor
LCS	= Launch Control System
ACL	= Application Command Language
ASF	= Application Services and Framework
C++	= Name of an Object Oriented programming Language

I. Abstract

The Launch Control System (LCS) developed at NASA's Kennedy Space Center (KSC) will be used to launch future spacecraft. Two of the many components of this system are the Application Control Language (ACL) and remote displays. ACL is a high level domain specific language that is used to write remote control applications for LCS. Remote displays are graphical user interfaces (GUIs) developed to display vehicle and Ground Support Equipment (GSE) data, they also provide the ability to send commands to control GSE and the vehicle. The remote displays and the control applications have many facets and this internship experience dealt with several of them.

II. Introduction

As mentioned before, ACL is part of the LCS developed at KSC, which will be used to launch future spacecraft. For remote control applications, the ASF created the Application Command Language (ACL). ACL is an abstract layer on C++ and is a language that is used by NASA engineers for checkout and launch of spacecraft. ACL provides NASA engineers the ability to monitor and control the vehicle and launch equipment prior to and during a launch. Due to the large amount of Application Programming Interface (API) calls thorough testing of the language is required.

The following report will explain what is involved in the process of writing, testing and documenting remote control software, unit testing for the ASF and ACL and other tasks involved in the LCS software development.

III. Remote Control Displays Creation

To be able to access the Display Editor (DE) for creating remote displays, one needs a Linux virtual machine to access it. The DE is a drag and drop environment, one can easily choose the type of object that wants to add and edit its properties to tie it to data and end items. These objects are called “Symbols” in the editor; examples of these are the Text Measurement which deals with text displays, the Command button to set values or change hardware states, the Display button that launches additional displays, and the State Component that displays images corresponding to enumeration states.

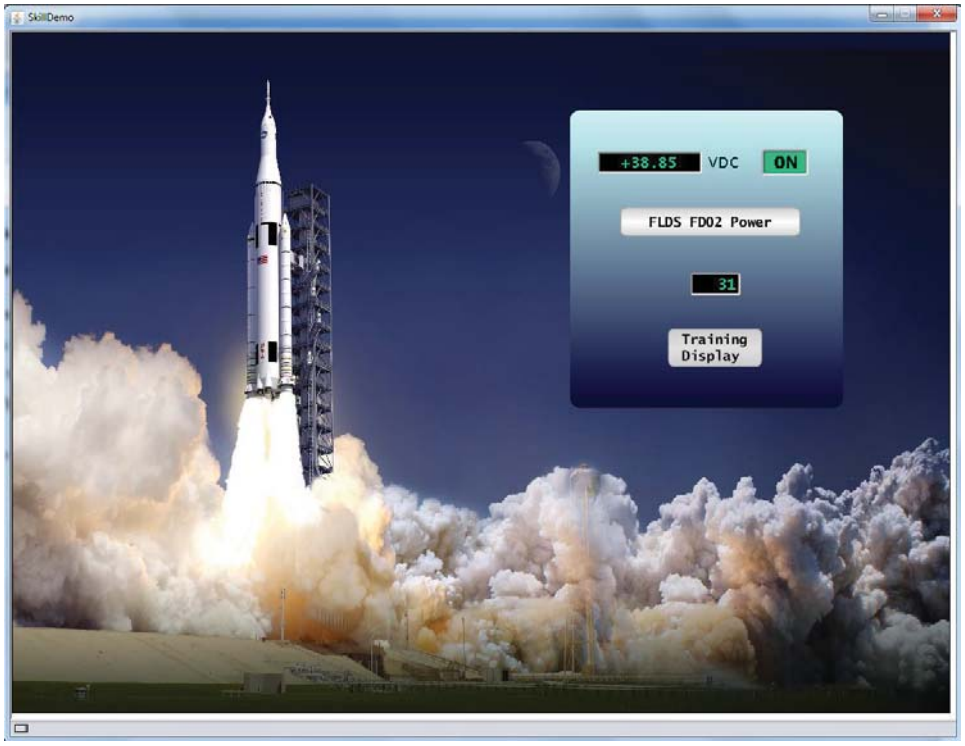
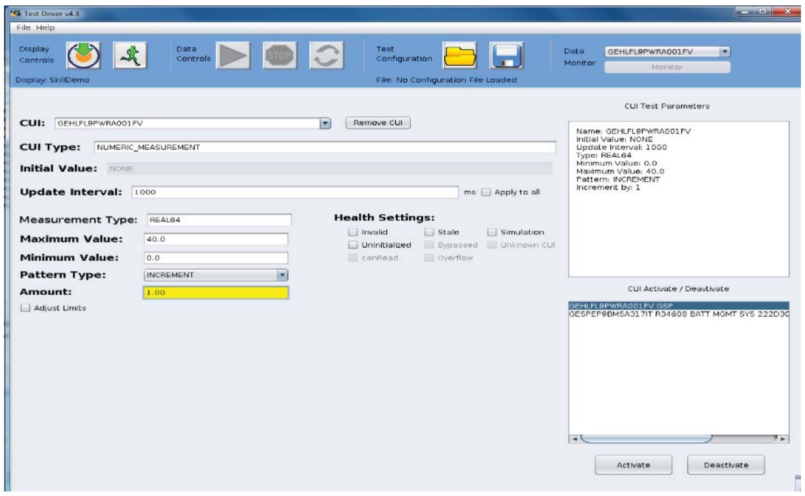


Image 1 – Example of a remote control display that tests each of the symbols and adds a background image.

When one adds a symbol to a display within the display editor, one has the ability to choose a Compact Unique Identifier (CUI). A CUI is an identifier that can represent measurement data and commands used within displays. Additionally within the LCS, CUIs are a method to be able to represent commands, measurements, events or prompts. CUIs provide the engineers with the ability to tie displays within the control system.

After symbols and images are added to the display one can run the display from the editor in the Test Driver. Measurement CUI values can be changed within the test driver to test how the display will look with different values. Eventually if one chooses to run the display it will work as expected if everything is correct. If we have a link to another display it will not work but it will show the path so that you know if the link is correct. Once a display is ready to be used it is then promoted to the AccuRev source control tool.



Display: SkillDemo
File: No Configuration File Loaded
Data Monitor: GEHLFL9PWRA001FV

CUI: GEHLFL9PWRA001FV
CUI Type: NUMERIC_MEASUREMENT
Initial Value: NONE
Update Interval: 1000 ms ☐ Apply to all

Measurement Type: REAL64
Maximum Value: 40.0
Minimum Value: 0.0
Pattern Type: INCREMENT
Amount: 1.00
☐ Adjust Limits

Health Settings:
☐ Invalid ☐ Stale ☐ Simulation
☐ Uninitialized ☐ Bypassed ☐ Unknown CUI
☐ canRead ☐ Overflow

CUI Test Parameters
Name: GEHLFL9PWRA001FV
Initial Value: NONE
Update Interval: 1000
Type: REAL64
Minimum Value: 0.0
Maximum Value: 40.0
Pattern: INCREMENT
Increment by: 1

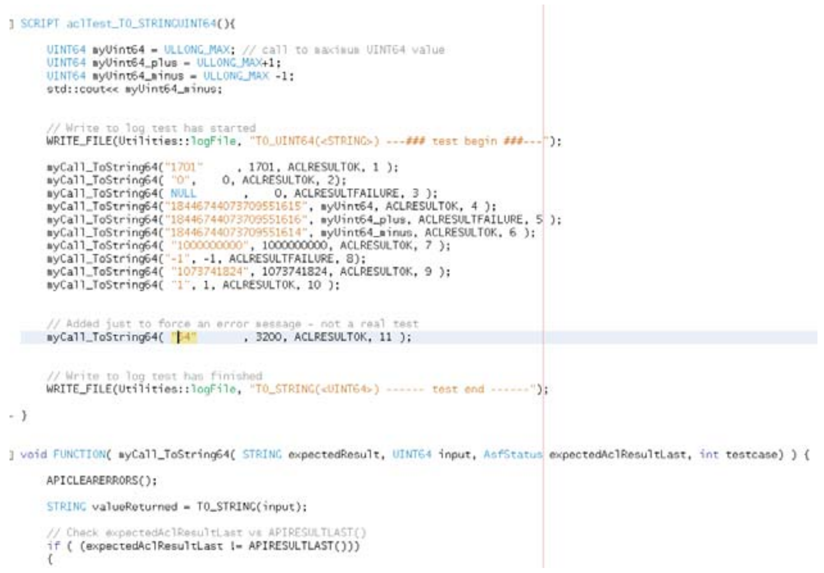
CUI Activate / Deavtivate
GEHLFL9PWRA001FV GSP
GESPEP9BMSA317IT R34608 BATT MGMT SYS 222D3C

Image 2 – Test display in the Test Driver, it shows the change one has to make (in yellow) if a float is used.

IV. ASF Functional Testing

In this internship, one was trained as a remote software developer for the Integrated Launch Operations Applications (ILOA) software, this is a software that is written in ACL. ILOA software is utilized by GSE subsystems; each subsystem focuses on a certain part of the rocket and ground components that the LCS needs.

Scripts done in ACL may also be used to initiate values shown on remote displays. To create a control application one has to log in to the Linux virtual machine environment and utilize the NetBeans Integrated Development Environment (IDE). After the control application is completed, one has to upload it to the AccuRev Software too.

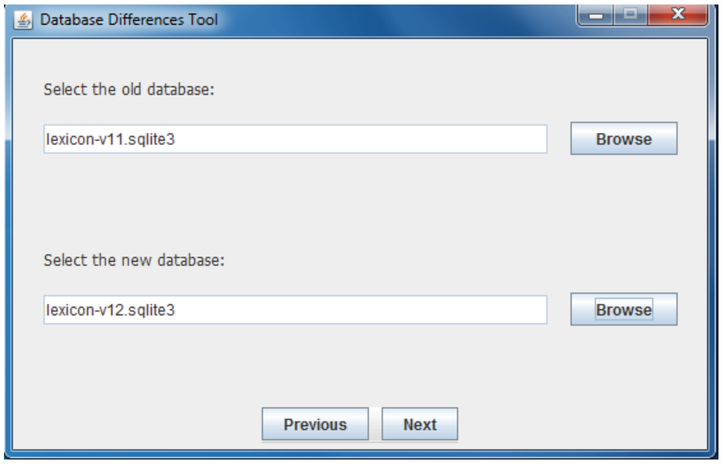


```
] SCRIPT aclTest_TO_STRINGUINT64(){

      UINT64 myUint64 = ULLONG_MAX; // call to maximum UINT64 value
      UINT64 myUint64_plus = ULLONG_MAX+1;
      UINT64 myUint64_minus = ULLONG_MAX -1;
      std::cout<< myUint64_minus;


      // Write to log test has started
      WRITE_FILE(Utilities::logFile, "TO_UINT64(<STRING>) ---### test begin ###---");

      myCall_ToString64("1701"     , 1701, ACLRESULTOK, 1 );
      myCall_ToString64( "0",    0, ACLRESULTOK, 2);
      myCall_ToString64( NULL        ,    0, ACLRESULTFAILURE, 3 );
      myCall_ToString64("18446744073709551615", myUint64, ACLRESULTOK, 4 );
      myCall_ToString64("18446744073709551616", myUint64_plus, ACLRESULTFAILURE, 5 );
      myCall_ToString64("18446744073709551614", myUint64_minus, ACLRESULTOK, 6 );
      myCall_ToString64( "1000000000", 1000000000, ACLRESULTOK, 7 );
      myCall_ToString64("-1", -1, ACLRESULTFAILURE, 8);
      myCall_ToString64( "1073741824", 1073741824, ACLRESULTOK, 9 );
      myCall_ToString64( "1", 1, ACLRESULTOK, 10 );


      // Added just to force an error message - not a real test
      myCall_ToString64( "|64"        , 3200, ACLRESULTOK, 11 );


      // Write to log test has finished
      WRITE_FILE(Utilities::logFile, "TO_STRING(<UINT64>) ------ test end ------");

- }

] void FUNCTION( myCall_ToString64( STRING expectedResult, UINT64 input, AsfStatus expectedAclResultLast, int testcase) ) {

      APICLEARERRORS();

      STRING valueReturned = TO_STRING(input);

      // Check expectedAclResultLast vs APIRESULTLAST()
      if ( (expectedAclResultLast != APIRESULTLAST()))
      {
```

Image 3 – ACL control application example.

V. Modifications to the Database Differences Comparison Tool

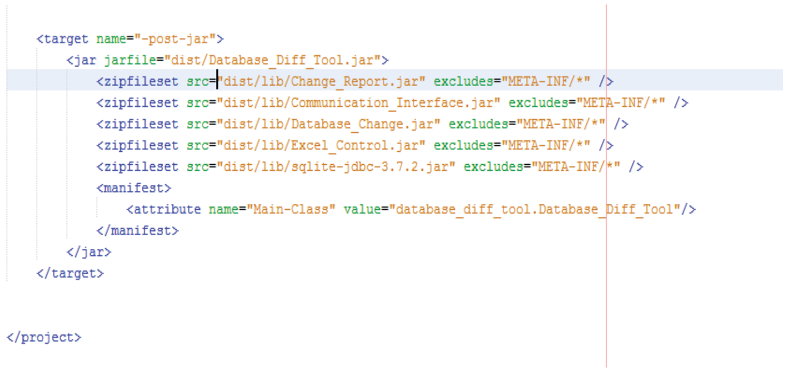
In addition to creating remote displays and control applications I made modifications to the Database Differences Comparison Tool (DDCT). It compares an old and a new CUI database and presents the changes in a log file. It was requested that new features should be added to the Graphical User Interface (GUI) and additional functionality be improved upon. First, the program was changed so that it remembers the file path of the databases, in an effort to reduce the number of mouse clicks by the user -both databases are now selected in the same window. Second, I improved the text file output format for the log file, by removing and unnecessary pop up and cleaning the log file output.



Select the old database: lexicon-v11.sqlite3

Select the new database: lexicon-v12.sqlite3

Image 4 – GUI modification example.



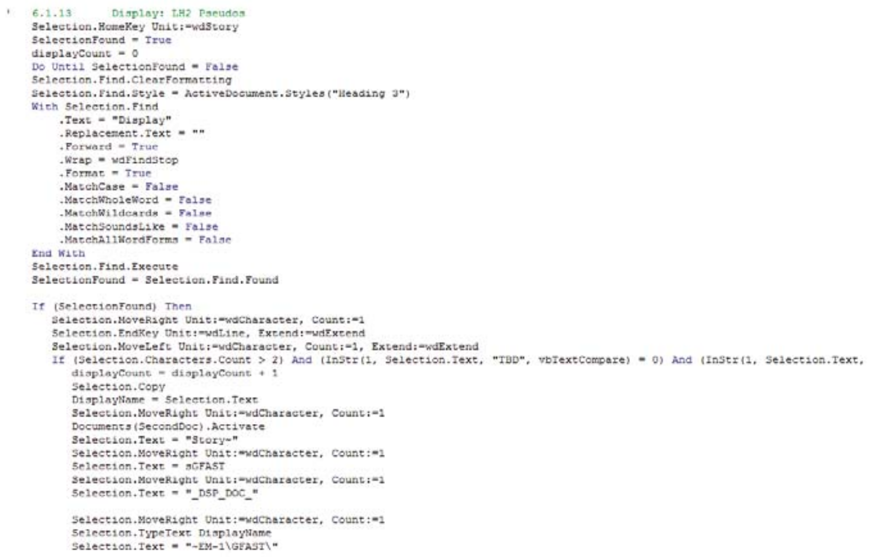
```
    <target name="-post-jar">
        <jar jarfile="dist/Database_Diff_Tool.jar">
            <zipfileset src="dist/lib/Change_Report.jar" excludes="META-INF/*" />
            <zipfileset src="dist/lib/Communication_Interface.jar" excludes="META-INF/*" />
            <zipfileset src="dist/lib/Database_Change.jar" excludes="META-INF/*" />
            <zipfileset src="dist/lib/Excel_Control.jar" excludes="META-INF/*" />
            <zipfileset src="dist/lib/sqlite-jdbc-3.7.2.jar" excludes="META-INF/*" />
            <manifest>
                <attribute name="Main-Class" value="database_diff_tool.Database_Diff_Tool"/>
            </manifest>
        </jar>
    </target>


</project>
```

Image 5 – Build.xml file modifications

Third, there was a problem in compilation in which the program needed several individual projects to run, so it could not be run as an executable file. To fix this, one has to modify the build.xml file of the main project. To include the other project files used- this ensures everything is included in one executable and can run on its own.

VI. Ground Flight Application Software Team (GFAST) Spreadsheets for Remote Displays Creation.

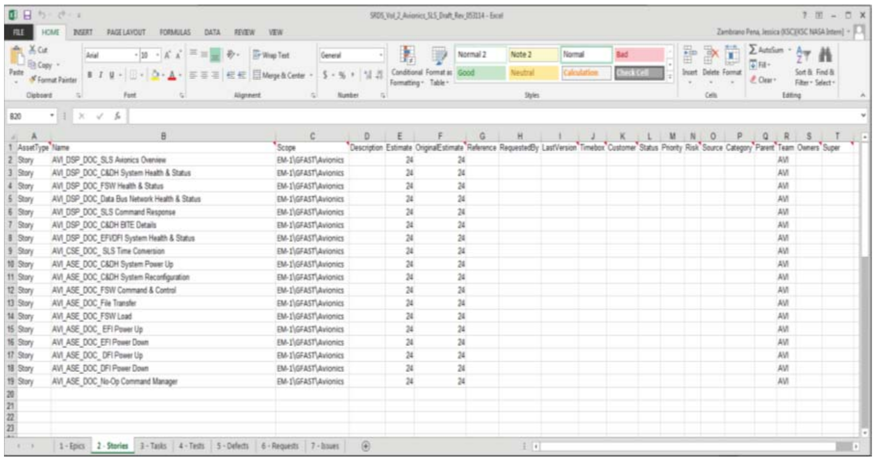
Along with GSE subsystem creating remote displays and control applications, they have to create a Software Requirements & Design Specification (SRDS) document. This document is then processed by the GFAST and imported into an excel spreadsheet. During the internship one created a Visual Basic macro that reads the SRDS documents and can export permanent data in an excel spreadsheet. The macro runs automatically and searches through the whole document for the text selected, which is the title of every section. The search has the capability to identify duplicate data within the document and thus not export duplicates to the spreadsheet. This search was implemented so that one can use extra parameters to avoid adding unfinished sections (avoiding everything that has TBD or the placeholder [name] in the name of the search sections).



```
'   6.1.13      Display: LH2 Pseudos
    Selection.HomeKey Unit:=wdStory
    SelectionFound = True
    displayCount = 0
    Do Until SelectionFound = False
    Selection.Find.ClearFormatting
    Selection.Find.Style = ActiveDocument.Styles("Heading 3")
    With Selection.Find
        .Text = "Display"
        .Replacement.Text = ""
        .Forward = True
        .Wrap = wdFindStop
        .Format = True
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = False
        .MatchSoundsLike = False
        .MatchAllWordForms = False
    End With
    Selection.Find.Execute
    SelectionFound = Selection.Find.Found

    If (SelectionFound) Then
       Selection.MoveRight Unit:=wdCharacter, Count:=1
       Selection.EndKey Unit:=wdLine, Extend:=wdExtend
       Selection.MoveLeft Unit:=wdCharacter, Count:=1, Extend:=wdExtend
       If (Selection.Characters.Count > 2) And (InStr(1, Selection.Text, "TBD", vbTextCompare) = 0) And (InStr(1, Selection.Text,
          displayCount = displayCount + 1
          Selection.Copy
          DisplayName = Selection.Text
          Selection.MoveRight Unit:=wdCharacter, Count:=1
          Documents(SecondDoc).Activate
          Selection.Text = "Story-"
          Selection.MoveRight Unit:=wdCharacter, Count:=1
          Selection.Text = sGFAST
          Selection.MoveRight Unit:=wdCharacter, Count:=1
          Selection.Text = "_DSP_DOC_"

          Selection.MoveRight Unit:=wdCharacter, Count:=1
          Selection.TypeText DisplayName
          Selection.Text = "-EM-1\GFAST\"
```

Image 6 – Code snippet from the search macro.

Another parameter implemented within the macro is the creation of spaces to change excel cells using the parameter “~”.There’s also a summary created that counts the amount of items in each section. Once the macro has run completely, the file summary is saved in two formats, one in a word format that contains the results and summary and the other in a delimited “.tsv” file (tilde delimited file). The tilde delimited file is the one that gets imported to excel, and after getting the delimitation (the tildes) erased, the file imports the data in the format specified for the spreadsheets.



AssetType	Name	Scope	Description	Estimate	OriginalEstimate	Reference	RequestedBy	LastVersion	Timebox	Customer	Status	Priority	Risk	Source	Category	Parent	Team	Owners	Super
Story	AVI_DSP_DOC_SLS Avionics Overview	EM-1\GFAST\Avionics		24	24												AVI		
Story	AVI_DSP_DOC_C&DH System Health & Status	EM-1\GFAST\Avionics		24	24												AVI		
Story	AVI_DSP_DOC_FSW Health & Status	EM-1\GFAST\Avionics		24	24												AVI		
Story	AVI_DSP_DOC_Data Bus Network Health & Status	EM-1\GFAST\Avionics		24	24												AVI		
Story	AVI_DSP_DOC_SLS Command Response	EM-1\GFAST\Avionics		24	24												AVI		
Story	AVI_DSP_DOC_C&DH BITE Details	EM-1\GFAST\Avionics		24	24												AVI		
Story	AVI_DSP_DOC_EFI/DFI System Health & Status	EM-1\GFAST\Avionics		24	24												AVI		
Story	AVI_CSE_DOC_ SLS Time Conversion	EM-1\GFAST\Avionics		24	24												AVI		
Story	AVI_ASE_DOC_C&DH System Power Up	EM-1\GFAST\Avionics		24	24												AVI		
Story	AVI_ASE_DOC_C&DH System Reconfiguration	EM-1\GFAST\Avionics		24	24												AVI		
Story	AVI_ASE_DOC_FSW Command & Control	EM-1\GFAST\Avionics		24	24												AVI		
Story	AVI_ASE_DOC_File Transfer	EM-1\GFAST\Avionics		24	24												AVI		
Story	AVI_ASE_DOC_FSW Load	EM-1\GFAST\Avionics		24	24												AVI		
Story	AVI_ASE_DOC_ EFI Power Up	EM-1\GFAST\Avionics		24	24												AVI		
Story	AVI_ASE_DOC_EFI Power Down	EM-1\GFAST\Avionics		24	24												AVI		
Story	AVI_ASE_DOC_ DFI Power Up	EM-1\GFAST\Avionics		24	24												AVI		
Story	AVI_ASE_DOC_DFI Power Down	EM-1\GFAST\Avionics		24	24												AVI		
Story	AVI_ASE_DOC_No-Op Command Manager	EM-1\GFAST\Avionics		24	24												AVI		

Image 7 – Example of an imported and delimited spreadsheet.

VII. Conclusion

The creation of remote displays and control applications is divided in many different facets. This software is part of the Launch Control System (LCS) developed at Kennedy Space Center (KSC) to launch future rockets and spacecraft. This report explained how one created a remote display and a control application. This report also covered the enhancements made to the Databases Differences Comparison Tool, and it covered the macro developed to create spreadsheets used for the documentation of GSE subsystem displays. This report detailed how one enhanced tools that are used to develop ILOA software for the launch control system.