INTRODUCTION

The test procedure approach discussed in this presentation applies to the Paging and Area Warning System (PAWS) Internet Protocol (IP) Version 2.0 an enhancement release intended for deployment within Kennedy Space Center.
BACKGROUND

• PAWS Client Application (PCA) GUI is the user’s interface to the system.

• PAWS Remote Application (PRA), RDBMS, and Asterisk VOIP PBX comprise the Central Component layer responsible for enabling PCAs to communicate with their configured end points (EP).

• SIP ATAs provide the digital to analog interface to the EPs.

• Most EPs are connected to two IP Networks for redundancy and failover.
RATIONAL

• The PCA systems integration, manual regression test cases were written in a object-oriented script-like language.

• This approach was taken since the PCA is a distributed, stateful, heavy Java client application that required some procedure steps to be executed simultaneously from multiple PCAs.

• Script-like notation was a more natural choice over prose given the technical nature of the application.
PCA GRAPHICAL USER INTERFACE
TEST FRAMEWORK

• The test framework provides another level of abstraction which simplifies the test case steps by reducing repetitiveness.

• The test framework allows test case steps to reference or “call” its services similar to invoking methods or functions in code.

• Implementing a test framework simplifies test case maintenance since most changes are made in the framework instead of in the test cases themselves.
TEST CASE DESIGN

• Top-down approach – test steps were written before the framework.

• Test verification pass/fail steps are performed in the test cases, not in the framework.

• Mini Spec sections included within the test procedure define the functional requirement(s). Comments in the test cases define the specific aspect of the requirement(s) being tested.
IMPLEMENTATION

The script-like notation contains conventions and keywords borrowed from many commercial automated test tools including classic Silk Test (4Test) and Mercury WinRunner. Some conventions and keywords were derived on the fly.

• Dot notation is used to reference object properties and methods.

• The “Using” and “this” keywords are used to establish the context.

• Three periods enclosed in square brackets: “MZ[…]” is the notation used to represent a subset or set of all meta-zones (MZs) within an Area.
**WEATHER PAGE OVERIDES CONCURRENT ADMINISTRATIVE PAGE TO SAME MZ.**

**TEST PROCEDURE**

Using PCA(Active) 
SetPanel(Panel LC-39)
SetPagingType(Admin)

Using PCA(JSTC) 
SetPanel(Panel LC-39)
SetPagingType(Weather)

Parallel 
Using PCA(Active) 
ActivatePage(PAD 398)

Using PCA(JSTC) 
ActivatePage(PAD 398)

**FRAMEWORK**

SetPanel(panel) {
PCA(this).Main Control Area.Panels.Select(panel);
}

SetPagingType(type) {
PCA(this).Paging Area.Paging Message Types.Select(type);
}

ActivatePage(MZ[...]) {
For each metazone in MZ PCA(this).Data Area.btnPush(metazone);
PCA(this).Paging Area.btnPush(activate);
PCA(this).Paging Area.Status Icon = <ACTIVE red bell>;
PCA(this).Paging Area.DEACTIVATE.IsEnabled = true;
}
TEST PROCEDURE

Using PCA(GenTwoStealth)
  
  - Waitfor : DialogBox--> Command Override
  - Verify : GetPagingStatus() = 'READY green clock'
  - Verify : GetIconTypes(PAD 39B) = 'non-symbol'
  - Verify : GetBackgroundColor(PAD 39B) = 'selectable'
  - Verify : GetPagingButtonStatus() = 'ACTIVATE', 'disabled'
  - Verify : For each MetaZone in {MZ[...]} : GetBackgroundColor(MetaZone) = 'selectable'

FRAMEWORK

GetPagingStatus() {
  /* ACTIVE red bell | READY green clock */
}

GetPagingButtonStatus() {
  /* ACTIVATE/DEACTIVATE | Enabled/Disabled */
  return (PCA(this).Paging.Area.Button.Properties(text, state));
}
WEATHER PAGE OVERRIDES CONCURRENT ADMINISTRATIVE PAGE TO SAME MZ. P. 3 OF 4

TEST PROCEDURE

Using PCA(JSTC)
- Verify: GetPagingStatus() = ‘ACTIVE red bell’
- Verify: GetBackgroundColor(PAD 39B) = ‘green’
- Verify: GetPagingButtonStatus() = ‘DEACTIVATE’, ‘enabled’
- Verify: For each MetaZone in (MZ[…] – PAD 39B):
  GetBackgroundColor(MetaZone) = ‘non-selectable’
- DeActivatePage()

FRAMEWORK

GetBackgroundColor(mz) {
/* selectable | non-selectable | green | red | orange | red slants */
return (PCA(this).Data Area.mz.GetProperty(Bkgrd Color));
}

DeActivatePage() {
PCA(this).Paging Area.btnPush(DEACTIVATE);
PCA(this).Paging Area.Status Icon = <READY green clock>;
PCA(this).Paging Area.ACTIVATE.IsEnabled = false;
}
**TEST PROCEDURE**

Using PCA(GenTwoStealth)
- Verify : GetPagingStatus() = ‘READY green clock’
- Verify : GetIconTypes(PAD 39B) = null
- Verify : For each MetaZone in (MZ[...]) :
  - GetBackgroundColor(MetaZone) = ‘selectable’

**FRAMEWORK**

GetIconTypes(mz) {
  /* busy-circle | green non-symbol */
  return (PCA(this).Data Area.mz.GetProperty(Icon Types));
}
SUMMARY

Anyone knowledgeable on the subject of automated software testing will recognize the approach described in this presentation as an exercise in automation without using an automated tool.

Many assumptions were made in the test framework concerning the actual object classes in the GUI. For example, it was assumed that the meta-zone object properties, e.g., background color and icon type, are exposed. However, the framework code would not be as straightforward if the meta-zones were implemented as discrete images.