ITEA Symposium 2015 – Automated Test for NASA CFS

David C. McComas[1], Susanne L. Strege[1],
Paul B. Carpenter[2], Randy Hartman[2]

[1] NASA Goddard Space Flight Center
[2] EXB Solutions, Inc. (EXB)

Medical

Aerospace

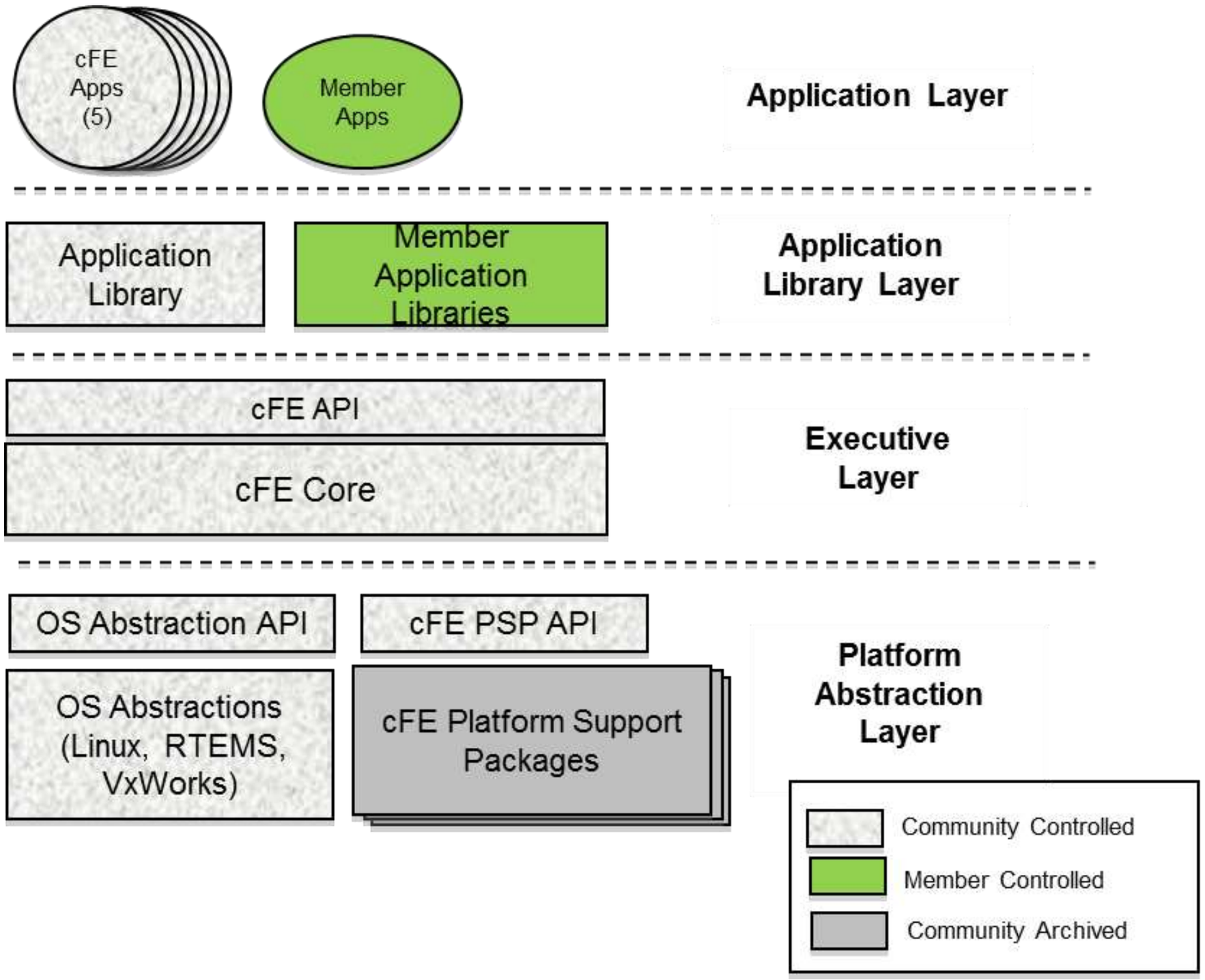
Government

Certified ISO 9001
Service-Disabled Veteran-Owned Small Business (SDVOSB)

core Flight System (cFS) Overview

- The core Flight System (cFS) is a flight software product line developed by the NASA Goddard Space Flight Center's Flight Software Systems Branch (FSSB) over the past 10 years.
- cFS is used in many NASA and commercial systems, such as:
  - Goddard Space Flight Center's Lunar Reconnaissance Orbiter (LRO)
  - Global Precipitation Measurement (GPM)
  - Magnetospheric Multi-scale Mission Spacecraft (MMS)
  - The Ames Research Center's Lunar and Dust Environment Explorer
  - Johnson Space Center's Morpheus project
- Previous cFS reuse efforts have not attained all of the benefits of reuse because they employed a "clone and own" approach.
- To improve upon the shortcomings of the "clone and own" approach, the FSSB now takes the entire cFS software development lifecycle into account and maintains reusable artifacts for each phase.
- To improve reuse, cFS has a layered architecture as shown on Slide 3.

cFS Layered Architecture



Objective of EXB Pilot Project

The objective of the pilot project is to apply EXB's Requirements-Based Testing Methodology and associated TestCompass® toolset on a set of cFS applications with the following goals in mind:

1. Demonstrate EXB's methodology and toolset to the cFS community while providing additional validation to the cFS applications themselves.

2. Evaluate EXB's methodology and toolset with respect to the cFS configuration space verification challenge and to the cFS test maintainability challenge.

3. Evaluate EXB's methodology and toolset as it is used on an application that is being matured for inclusion into the cFS app library.

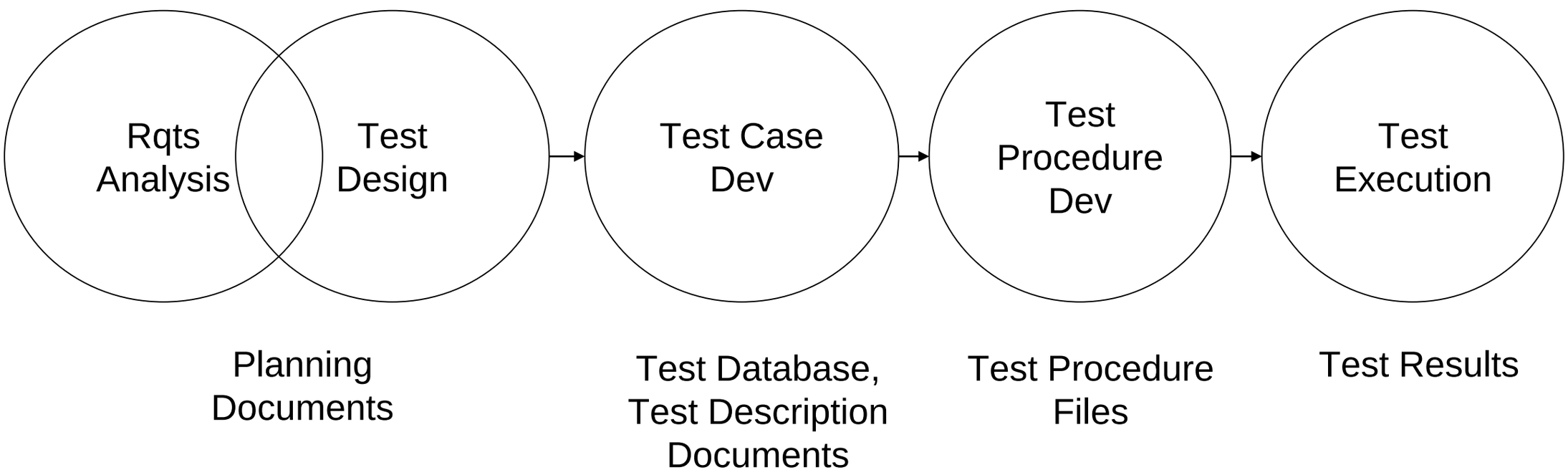
- Note: Apps that are originally designed for a single mission or under a technology effort need to undergo a maturation process before they are suitable for Class B missions and compliant with the cFS product line standards.

EXB Requirement-Based Testing Methodology

TestCompass® automates five areas of requirements-based testing:

1. Requirements Analysis
2. Test Design
3. Test Case Development
4. Test Procedure Development
5. Test Execution

EXB's TestCompass® toolset automates EXB's methodology.

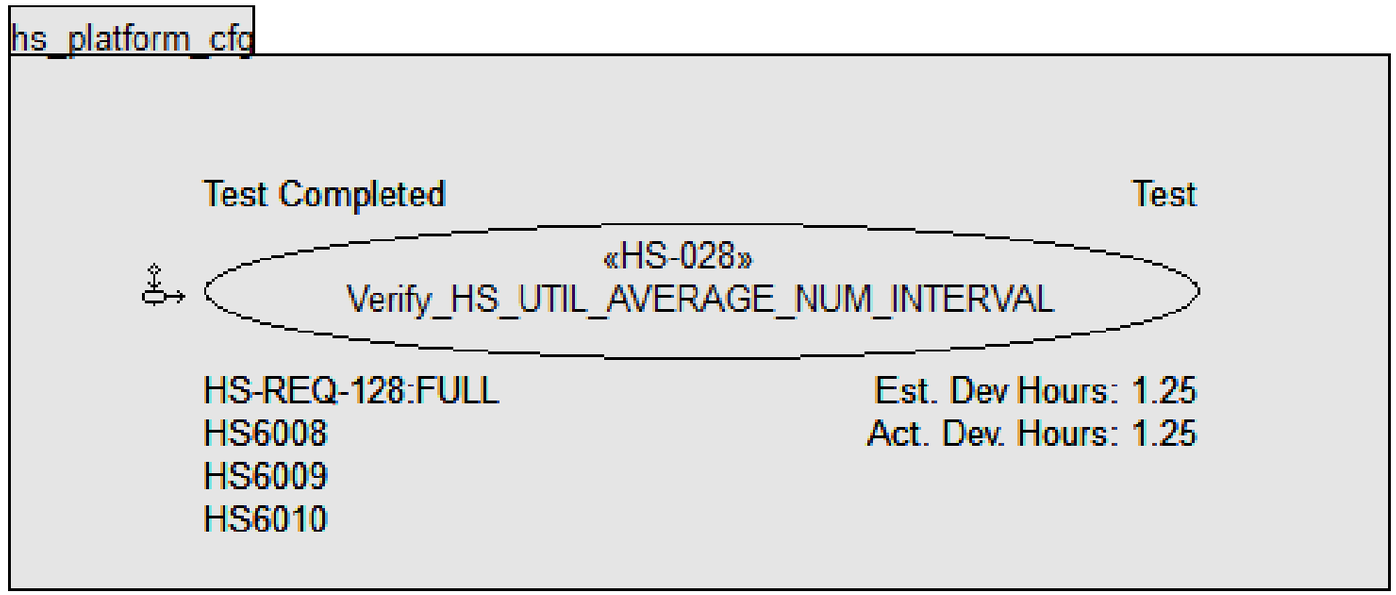


Reports: Status, Cost Index, Traceability

Requirements Analysis and Test Design

- Requirements analysis provides:
  - Testable requirements
  - Initial project schedule and scope
  - Initial coverage analysis
  - Rapid impact of requirement changes

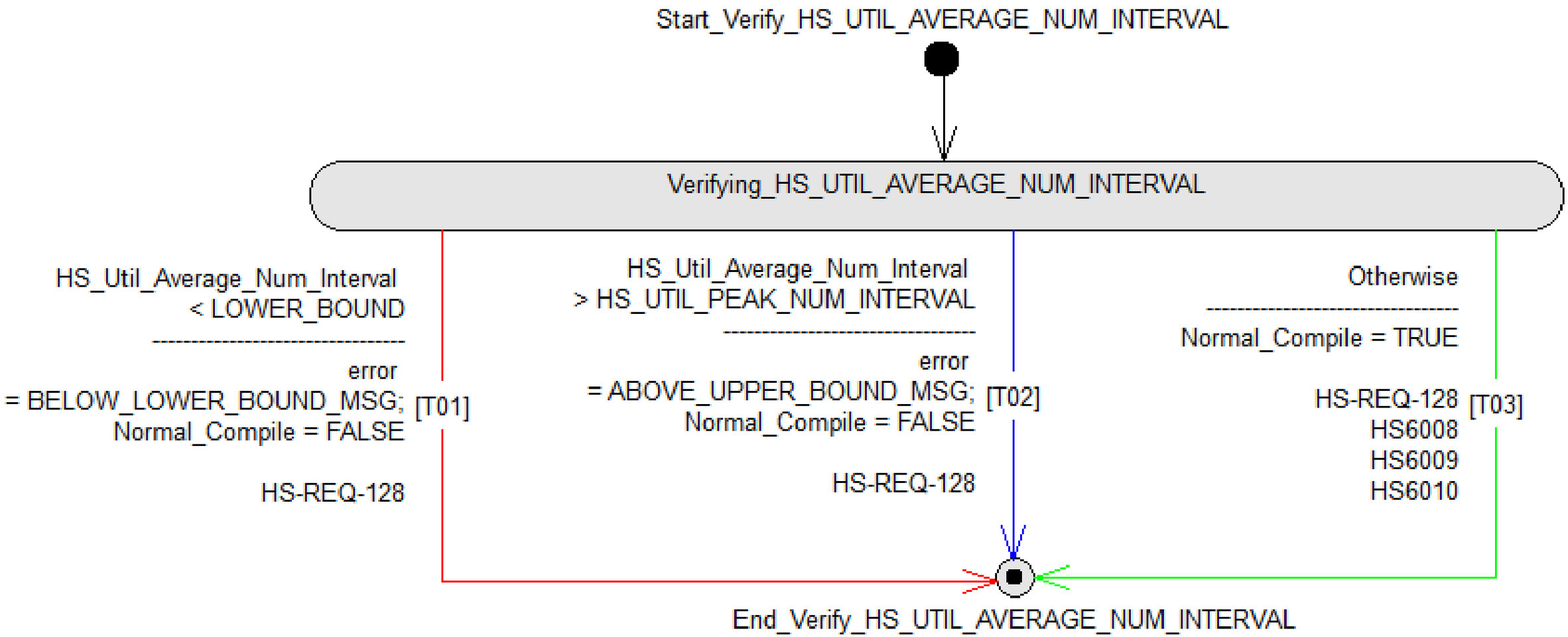
- UML Use Cases capture:
  - Test name and Id
  - Requirements Trace
  - Status



Test Case Development

Tests are modeled with UML Activity Diagrams.
The diagrams capture:

- Test scenarios
- Test behavior
- Expected outputs
- Requirements traceability



Test Case Development - Test Data Samples

- Based on the criteria specified in the Test Database, TestCompass automatically generates Test Data Samples.

Data Type: T_HS_Util_Average_Num_Interval

Base Type: int

Purpose	Value	Robustness
Reference	4	Normal
Minimum Below Bounds	0	Robust
Minimum Value	1	Normal
Minimum Debug	2	Normal
Maximum Debug	63	Normal
Maximum	64	Normal
Maximum Above Bounds	65	Robust

TestCompass combines the samples into Test Cases as shown on Slide 9.

MCDC can be defined

Test Case Development - Test Cases

Test Name: Verify_HS_UTIL_AVERAGE_NUM_INTERVAL

Test Scenario: Verify_HS_UTIL_AVERAGE_NUM_INTERVAL_Below_Bounds

Case ID	Robustness	Inputs	Expected Outputs
HS-028_1_0001	Robust	HS_Util_Average_Num_Interval = 0	error = "HS_UTIL_AVERAGE_NUM_INTERVAL cannot be less than 1", Normal_Compile = FALSE

Test Scenario: Verify_HS_UTIL_AVERAGE_NUM_INTERVAL_Above_Bounds

Case ID	Robustness	Inputs	Expected Outputs
HS-028_2_0001	Robust	HS_Util_Average_Num_Interval = 65	error = "HS_UTIL_AVERAGE_NUM_INTERVAL can not exceed HS_UTIL_PEAK_NUM_INTERVAL", Normal_Compile = FALSE

Test Scenario: Verify_HS_UTIL_AVERAGE_NUM_INTERVAL_In_Bounds

Case ID	Robustness	Inputs	Expected Outputs
HS-028_3_0001	Normal	HS_Util_Average_Num_Interval = 4	Normal_Compile = TRUE
HS-028_3_0002	Normal	HS_Util_Average_Num_Interval = 1	Normal_Compile = TRUE
HS-028_3_0003	Normal	HS_Util_Average_Num_Interval = 2	Normal_Compile = TRUE
HS-028_3_0004	Normal	HS_Util_Average_Num_Interval = 63	Normal_Compile = TRUE
HS-028_3_0005	Normal	HS_Util_Average_Num_Interval = 64	Normal_Compile = TRUE

Test Procedure Development

- This is an automated process using TestCompass and project-specific software.
- The project-specific software generates test procedures in client-specific software testing languages.
- EXB develops the test procedure generator with guidance from the client.
- When testing high-level software requirements, the test procedure generator creates test drivers, and library functions that are completed by the test engineers.
- When testing software design requirements using a language such as C, the majority of the test procedures can be fully generated by the test procedure generator.
- For the cFS project, EXB generated header files to replace the default configuration parameter files. In this project, the compile step is automated as part of the Test Procedure Development.

Slide 11 shows the result of executing Test Case HS-028_1_0001 from Slide 9.

Test Execution - Test Results

Test: Verify_HS_UTIL_AVERAGE_NUM_INTERVAL

Results Summary

P/F	Test Points
Pass	9
Fail	0
Total	9

TestScenario: Verify_HS_UTIL_AVERAGE_NUM_INTERVAL_Below_Bounds

Test ID: HS-028_1_0001

Inputs

Variable	Value
HS_Util_Average_Num_Interval	0

Outputs

Variable	Expected Value	Actual Value	Pass/Fail
error	"HS_UTIL_AVERAGE_NUM_INTERVAL cannot be less than 1"	"HS_UTIL_AVERAGE_NUM_INTERVAL cannot be less than 1"	PASS
Normal_Compile	FALSE	FALSE	PASS

Conclusions of pilot

- EXB's methodology and toolset have been demonstrated to provide a well-defined repeatable process with artifacts suitable for long term maintenance. Therefore this approach may serve as a common cFS application verification method.

- The next step is to continue to apply EXB's methodology and toolset to additional cFS applications and to determine whether it will serve as the standard cFS verification methodology.

Summary

- Using an approach as demonstrated in this pilot could lead to more reuse of software in safety critical applications
- Cost efficiencies gained could lead to wider application adoption of proven software
- Potential for more rapid certification
- Potential for application certification

Chris Schwartzbauer
EXB Solutions Inc

chris.schwartzbauer@exbsolutions.com
+1 (612) 208-8110