

SCA Training for Developers and Testers



Day 5: SCA Porting, Test and Evaluation

Copyright © 2002, Raytheon Company.
All Rights Reserved

Day 5 AGENDA

- SCA Porting
- SCA Test and Evaluation
- Open SCA Issues

JTRS

SCA Porting

SCA Porting

- Porting Definitions
- Porting Planning
- Porting Successes / Lessons Learned

What is Porting?

- Porting is the process of transferring software applications (core framework, services, waveforms) and/or hardware devices (with their associated Logical Device, device drivers, XML profile) onto a different hardware platform
- Ideally, porting onto a different platform does not entail need for any software modification
 - this does not happen in the real world
- So, successful porting is the ability to transfer software and hardware to a different platform with 'minimal' redesign / recoding

CF, Service, Device Porting

- In order to achieve operating efficiency, the CF, basic services (e.g. Time), and hardware devices have direct coupling to the basic radio platform - they provide HW-SW isolation to (waveform) software applications
- The overriding requirement for use of standard interfaces, and the inclusion of APIs in the SCA, will allow for these elements to be ported
- Some hardware-, OS-, or ORB-dependent software will require modification
- XML profiles may need modification

Application Porting

- The SCA is aimed at minimizing the need for modifying application software when porting
- Elements of application porting:
 - recompile code written for one processor to run on another
 - recode functions, written in a higher order language, into VHDL due to the radio architecture putting them in an EPLD/FPGA
 - possibly modify code to account for the development tools of a different operating system
 - rewrite hardware-dependent code, i.e. where assembly language routines may be necessary
 - hardware-dependent code should be implemented using SCA-defined device and I/O APIs where possible

Porting Planning

- Steps for successful porting
 - Radio architecture knowledge transfer
 - Porting-application knowledge transfer
 - Concurrence on port methodology
 - Development
 - Integration and test
- Assumptions
 - Radio and application are SCA compliant
 - Contractual vehicle with appropriate Proprietary Information Agreement(s) in place

Knowledge Transfer

- Radio processing implementation (general purpose processors / digital signal processors / electronically programmable logic devices)
- Available capacity, memory, etc.
- Available devices & services using SCA-defined APIs

- Application partitioning constraints / implementation language(s)
- Documentation of WF components, XML, message flows
- Performance needs (timing / latency / etc.)
- Security considerations

Porting Methodology

- Who should port? Depends on several factors:
 - new waveform => waveform designer?
 - proven waveform from JPO library => HW or system provider?
 - tightly constrained hardware implementation => HW provider?
 - contractual limitations may be deciding factor
 - other?
- Best chance of success (schedule, budget, performance) if use IPT structure
 - continued involvement (and commitment) of hardware contractor, application contractor, government customer
 - clearly identified responsibilities at start

- Generate detailed plan
 - identify facilities and tools
 - integration plan
 - individual components
 - intra-component communication on new HW platform
 - component to new HW devices / services
 - external interconnections (GUI, RF) and end-to-end functionality
 - milestone schedules
 - documentation requirements
 - reviews

Integration and Test

- Location and responsibility determined earlier; typically at hardware vendor's or system integrator's facility due to availability of support equipment and test assets
- Test for proper operation / interoperability of ported application
- Test previous capabilities of radio with new application running
- Include validation of continued SCA compliance, per JPO process

Porting Successes

- As part of the JTRS Step 2A program, porting was successfully accomplished
 - Core Framework, developed by Raytheon, was ported to ITT and Rockwell-Collins radios
 - HF-ALE waveform developed by Rockwell was ported to Raytheon radio
 - VHF FM waveform developed by Raytheon was ported to ITT and Rockwell radios
- JTRS Step 2B porting
 - Raytheon Core Framework ported to Thales handheld radio
 - SINCGARS ESIP/INC waveform developed by Assurance Technology Corporation being ported to Raytheon radio

Porting Lessons Learned

- COTS (sometimes) means you may not get everything you expect
 - full compliance to industry standards not rigorously enforced
- Different processors, OSs, ORBs have unique naming conventions, data formatting, operations, etc. that must be taken into account; examples:
 - Load and Execute operations for applications differ between OSs, e.g. LynxOS and VxWorks
 - Identical variables from the symbol tables of an x86 and PPC use a different naming convention; e.g. (PPC) “A” = (x86) “_A”
 - ORB vendors use proprietary bind operations that provide many possible combinations of object lookups; the documentation does not cover the list of allowable combinations
 - All CORBA ORB vendors provide exception handling macros that are NOT portable
 - (these differences mainly affect porting CF, Devices, Services)

Porting Lessons Learned, cont'd

- Inclusion of API and Service definitions in SCA will further enhance porting efficiency
- Porting effort planning and continued dialogue between parties is critical to success
- Phased approach creates reasonable increments of porting scope and enables isolation of encountered problems
 - initial port to representative target hardware, without other processes / channels operating
 - then port to actual target hardware

SCA Test and Evaluation

Validation Overview

- Validation plan covers the validation and verification of the SCA Core Framework/ Waveform requirements.
- Validation Plan Highlights:
 - Mapped requirements from analysis model to tests
 - System-level tests for majority of requirements
 - Validation procedures
 - Validation deliverables
- Goals:
 - Validate each CF/WF implementation against the requirements and reduce validation costs via automated tools and test “scripts”
 - Validate the WF against legacy requirements/ equipment for functionality.
 - Validate the WF against Standard OE for SCA-Compliance.
 - Validate the WF against SCA-defined APIs as necessary.

Coverage/Compliance Verification

- Create a requirements database containing files of SCA and WF requirements.
 - Requirements entered with attributes in accordance with the validation plan. The attributes allow the requirements of the SCA to be associated within the database.
 - These “linkages” form the flowdown from the WF analysis model and SCA requirements to the System/Unit Test requirements.
 - The database will be capable of being filtered by attribute to provide such validation results as the amount of SCA coverage achieved.
 - The interim and final validation reports will refer to the database and draw conclusions as to the effectiveness of the validation by reviewing the database.

Coverage/Compliance Verification

- Verification Procedure is prepared for each requirement or set of related requirements in the compliance matrix.
- Traceability between the analysis model and the implemented CF/WF provided to enable verification of a correct implementation.
 - Validation of requirements
 - That the primary WF analysis model and SCA requirements are adequately covered by the prototype implementation
 - That the prototype correctly implements these primary requirements
 - That each WF correctly implements the WF functionality.
 - That each WF implements APIs using SCA Building Blocks in keeping with the spirit of the SCA.

Coverage/Compliance Verification

- Automated Testing
 - System-level testing for the majority of the requirements, and unit tests for unreachable paths to supplement system-level testing.
 - “Golden” WF Test Applications with corresponding XML
- Code Inspections
 - Review of documentation/code to establish conformance with SCA and APIs
 - Inspection description
 - Includes objective, type (HW/SW/etc.), and type of requirement
 - Inspection compliance criteria
 - Includes any criteria specified in the requirements specification
 - Inspection requirements
 - Includes an Identifier of the requirements, special requirements, assumptions and constraints such as anticipated limitations on the inspection due to system or conditions, equipment, personnel, database, safety, security, and privacy considerations.
 - Type of data to be recorded.
 - Type of data recording/reduction/analysis to be employed

Validation Plan

- The Validation Plan details the requirements verification of the implementation.
- The Validation Plan describes:
 - Validation Methods
 - SCA Coverage/Compliance Verification
 - Verification Process
 - Validation Roles and Responsibilities
 - Schedule
 - SQA Roles
 - Verification Procedure for each requirement in the compliance matrix.

Validation Procedure

- The Validation Test Procedures ensure the implementation
- Satisfies the SCA and analysis model requirements
 - The objective for the requirement(s) being verified
 - The criteria for the requirement(s) being verified
 - Requirement identifier
 - Expected verification results
 - Evaluation method
 - Steps to show compliance to the requirement(s)

Validation Procedure

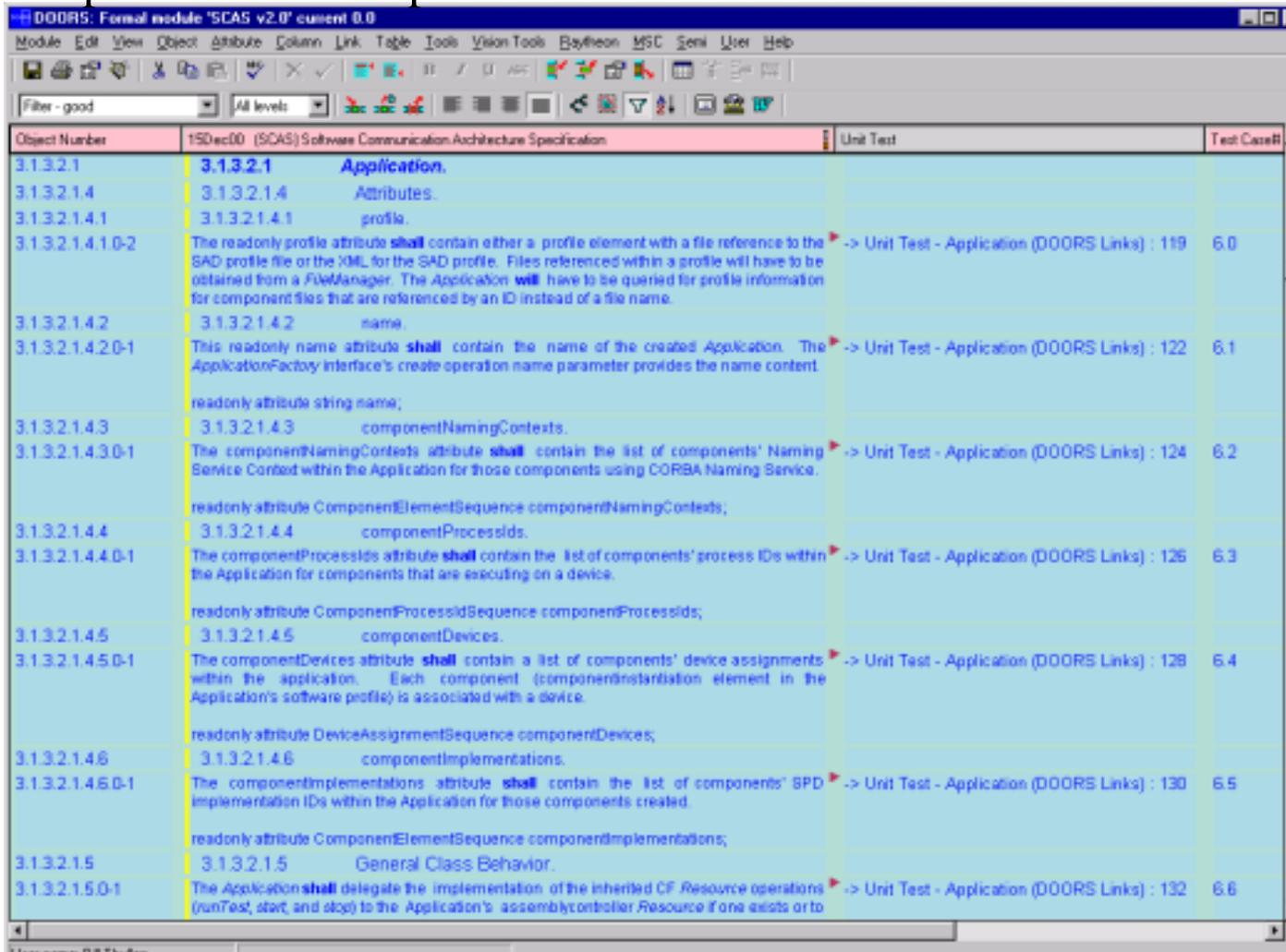
Mapping Requirements to Test Cases

- Requirements Management Tool (e.g.)
DOORS (Dynamic Object Oriented Requirements System)
 - Requirement Management Tool
 - Models Object-Oriented Methodology
 - Requirements modeled as discrete objects
 - Used for requirements database tracking

Validation Procedure

Mapping Requirements to Test Cases

Sample DOORS Output:



Object Number	15Dec00 (SCAS) Software Communication Architecture Specification	Unit Test	Test Case#
3.1.3.2.1	3.1.3.2.1 Application.		
3.1.3.2.1.4	3.1.3.2.1.4 Attributes.		
3.1.3.2.1.4.1	3.1.3.2.1.4.1 profile.		
3.1.3.2.1.4.1.0-2	The readonly profile attribute shall contain either a profile element with a file reference to the SAD profile file or the XML for the SAD profile. Files referenced within a profile will have to be obtained from a FileManager. The Application will have to be queried for profile information for component files that are referenced by an ID instead of a file name.	-> Unit Test - Application (DOORS Links) : 119	6.0
3.1.3.2.1.4.2	3.1.3.2.1.4.2 name.		
3.1.3.2.1.4.2.0-1	This readonly name attribute shall contain the name of the created Application. The ApplicationFactory interface's create operation name parameter provides the name content. readonly attribute string name;	-> Unit Test - Application (DOORS Links) : 122	6.1
3.1.3.2.1.4.3	3.1.3.2.1.4.3 componentNamingContexts.		
3.1.3.2.1.4.3.0-1	The componentNamingContexts attribute shall contain the list of components' Naming Service Context within the Application for those components using CORBA Naming Service. readonly attribute ComponentElementSequence componentNamingContexts;	-> Unit Test - Application (DOORS Links) : 124	6.2
3.1.3.2.1.4.4	3.1.3.2.1.4.4 componentProcessIds.		
3.1.3.2.1.4.4.0-1	The componentProcessIds attribute shall contain the list of components' process IDs within the Application for components that are executing on a device. readonly attribute ComponentProcessIdSequence componentProcessIds;	-> Unit Test - Application (DOORS Links) : 126	6.3
3.1.3.2.1.4.5	3.1.3.2.1.4.5 componentDevices.		
3.1.3.2.1.4.5.0-1	The componentDevices attribute shall contain a list of components' device assignments within the application. Each component (componentInstallation element in the Application's software profile) is associated with a device. readonly attribute DeviceAssignmentSequence componentDevices;	-> Unit Test - Application (DOORS Links) : 128	6.4
3.1.3.2.1.4.6	3.1.3.2.1.4.6 componentImplementations.		
3.1.3.2.1.4.6.0-1	The componentImplementations attribute shall contain the list of components' SPD implementation IDs within the Application for those components created. readonly attribute ComponentElementSequence componentImplementations;	-> Unit Test - Application (DOORS Links) : 130	6.5
3.1.3.2.1.5	3.1.3.2.1.5 General Class Behavior.		
3.1.3.2.1.5.0-1	The Application shall delegate the implementation of the inherited CF Resource operations (runTest, start, and stop) to the Application's assemblycontroller Resource if one exists or to	-> Unit Test - Application (DOORS Links) : 132	6.6

Validation Procedure

- Validation Methods
 - Test requirements with witness (JPO/SQA)
 - Witness to sign off test result
 - Problems encountered will be documented with SPR
 - SPR resolution procedure

Validation Procedure

- CF FileManager Test Procedure example:

FileManager: Operations						
	TEST/ METHOD:	INPUT	EXPECTED RESULTS	TEST CASE#	<u>Pass</u>	<u>Fail</u>
	getMounts()	Invoke Method	Return - string mountPoint "Hard_Drive" Return - FileSystem fs "true"	12.5	<input type="checkbox"/>	<input type="checkbox"/>
	unmount()	Enter string mountPoint " Hard_Drive " Invoke Method	No Return - no exception error occurs	12.4	<input type="checkbox"/>	<input type="checkbox"/>
	mount()	Enter string mountPoint " Hard_Drive " Copy FileSystem fs to FileSystem file_System Invoke Method	No Return - no exception error occurs	12.3	<input type="checkbox"/>	<input type="checkbox"/>
	getMounts()	Select getMounts - Invoke Method	Return - string mountPoint "Hard_Drive" Return - FileSystem fs "true"	12.5	<input type="checkbox"/>	<input type="checkbox"/>
	query()	Highlite FileSystem fs - Open Select query() method Enter DataType string id "SIZE" and "AVAILABLE_SPACE - Invoke Method	No Return - no exception error occurs - query setup Returns - DataType any value "2199617024" Returns - DataType any value "2183502848"	12.7	<input type="checkbox"/>	<input type="checkbox"/>

Test and Demonstration Process

- Provides repeatable evidence of correctness by exercising the item to verify satisfaction of requirements.
- Demonstrates the item implementation performs its intended functions and provides confidence that the implemented item does not perform unintended functions.
- Procedures documented in sufficient detail so that a second party could reproduce the test results.
- Apply the following process for each test/demonstration :
 1. Development of test and demonstration procedures
 2. Review of test cases/procedures
 3. Dry runs of test cases (informal event)
 4. Problem report tracking
 5. Formal Testing and Demonstration
 6. Regression testing (if required)
 7. Validation report

Test Cases/Procedures

- Reviews of the Test Cases/Procedures
 - Completed detailed evaluation procedures will have reviews conducted by the Lead Validation Engineer to ensure the test cases and evaluation procedures verify the intended coverage of the requirements.
- Dry Runs
 - Informal execution of all evaluation procedures from beginning to end.
 - Ensure the procedures, implementation, and test environment is ready to be executed formally. The lead Validation engineer and designees will conduct each dry run. The configuration of the dry run test equipment and prototype will be documented and controlled informally.
- Track Problem Reports (PRs)
 - Problems entered into software problem can correction report (SPR) tracking system.

Formal Testing and Demonstration

- Formal testing of the software will consist of executing the formal evaluation procedures.
- A Test Log will be maintained to document the test activity.
 - Test log will also be attached to the Validation Report (Validations Artifact documentation).
 - For each evaluation procedure executed, the procedure number, test personnel present, and overall procedure results will be recorded. Any discrepancies which occur during execution of the evaluation procedure will be recorded in the test log.
 - Formal test log data will be recorded during the formal verification. Formal test log data should contain, as a minimum, the following:
 - The version of the evaluation procedure used.
 - The version of the implementation component being verified.
 - The results of each verification including a pass or fail declaration.
 - The discrepancy between expected and actual results. (Problem Report)

Formal Testing and Demonstration

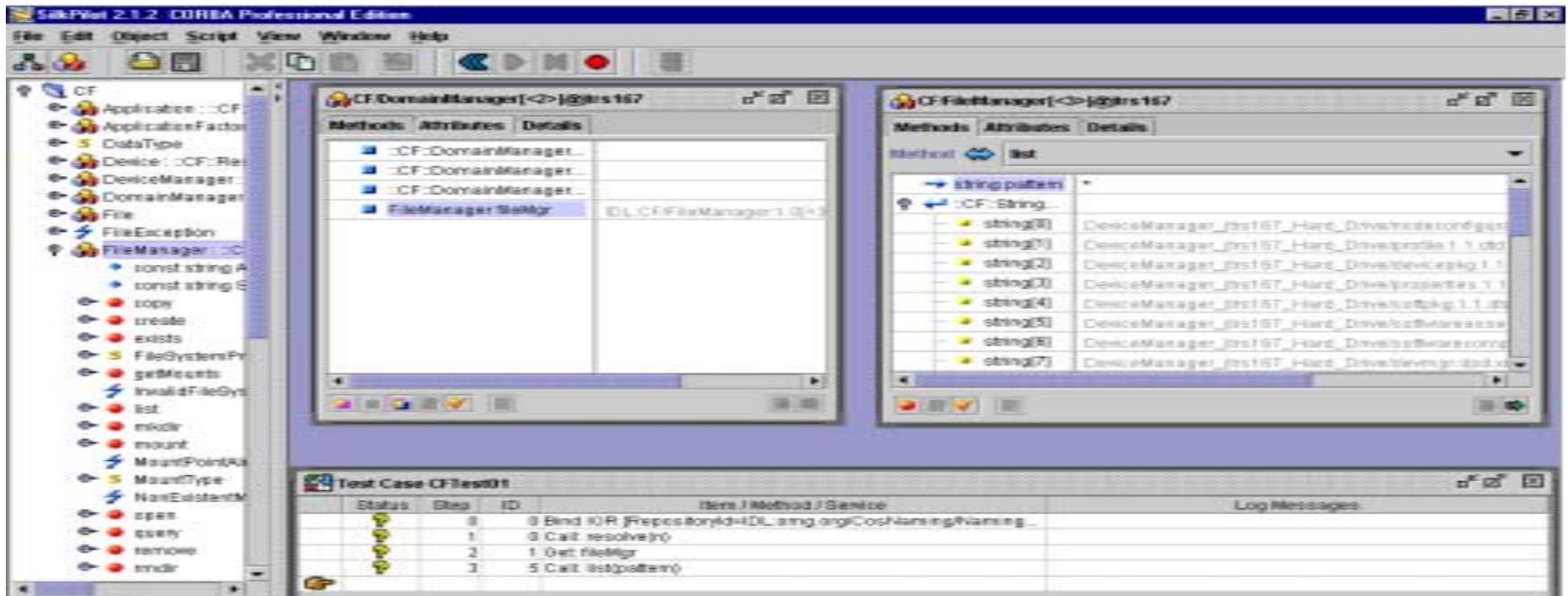
- Regression Testing
 - Consists of testing software corrections or changes made after the initial software release.
 - Re-executing the evaluation procedures and the operating manual defined for formal testing in the evaluation procedures.
- Validation Report (Accomplishment Summary)
 - Validation Team is responsible for the development of the Validation Report.
 - At a minimum the report will include some or all of the following; applicability, configuration index, development process including assumptions, validation matrix, supplemental analysis, test log data, problem reports, and validation conclusions.

Automated Validation Test Tools

- SilkPilot: Functional and regression testing of CORBA servers
 - Provides:
 - Interactive and automated testing of objects
 - Source code generation for test scripting
 - Playback of testing scripts
 - Stand-alone test clients
 - CORBA Interface browser
 - Inventory of objects and data structures
 - Java-based (available on NT and Unix)
 - Facilitates both the system-level and unit tests of the implemented software.
 - Vendor: Segue Software

Validation Test Tool

- Example of SilkPilot Usage



The screenshot displays the SilkPilot 2.1.2 CORBA Professional Edition interface. The left pane shows a tree view of objects, with 'FileManager' selected. The middle pane shows the 'Methods' tab for 'CF:DomainManager[<->]@jtr167', listing methods like 'FileManager\$FileMgr'. The right pane shows the 'Methods' tab for 'CF:FileManager[<->]@jtr167', listing methods like 'string(pattern)'. The bottom pane shows a 'Test Case CFtest01' with a table of test steps.

Status	Step	ID	Item / Method / Service	Log Messages
Pass	0	0	Bind IOR (RepositoryId=IDL:org/CosNaming/Naming...	
Pass	1	0	Call resolve()	
Pass	2	1	Get fileMgr	
Pass	3	5	Call list(pattern)	

Validation Test Tool

- Example SilkPilot Code Generation

```
/**
 * CREATOR: SilkPilot
 * VERSION: 2.1.2
 * TARGET ORB: VisiBroker
 */

import org.omg.CORBA.*;
import java.util.*;
import java.io.*;

public class Client
{
    ...
    public void run()
    {
        try {
            bindObject_0();
            call_resolve_0();
            get_fileMgr_1();
            call_list_2();
        } catch (org.omg.CORBA.SystemException ex) {
            handleSystemException(ex);
            _failed = true;
        }
        if (_failed == true) {
            System.out.println("\nTEST FAILED");
            System.exit(1);
        } else {
            System.out.println("\nTEST PASSED");
            System.exit(0);
        }
    }
}
```



Validation Metrics

- Example of SCA CF Requirements

Framework Control I/F:

3.1.3.2.1 Application:	33	
• 3.1.3.1.2 LifeCycle:		5
• 3.1.3.1.3 TestableObject:	3	
• 3.1.3.1.4 PropertySet:	8	
• 3.1.3.1.5 Resource:		<u>5</u>
Application Total:		54
3.1.3.2.2 ApplicationFactory Total:		36
3.1.3.2.3 DomainManager Total:		54
3.1.3.2.5 DeviceManager:		22
• 3.1.3.1.2 LifeCycle:		5
• 3.1.3.1.3 TestableObject:	3	
• 3.1.3.1.4 PropertySet:	8	
• 3.1.3.1.5 Resource:		5
• 3.1.3.2.4 Device:		<u>52</u>
DeviceManager Total:		95

Framework Services I/F:

3.1.3.3.1 FileTotal:	21	
3.1.3.3.2 FileSystem Total:		31
3.1.3.3.3 FileManager Total:		22
3.1.3.3.5 Logger Total:	<u>39</u>	

Total # Requirements: 352

Number of System and Unit Tests: TBD

Hardware Validation Overview

- “The primary purpose of the hardware structure is to *require complete and comprehensive publication of interfaces and attributes* once systems have been built.
- With these published specifications, *additional vendors can provide modules* within a system and *software developers can identify hardware modules with capabilities required* for a particular waveform application.
- Hardware modularity also *facilitates technology insertion* as future programmable elements increase in capability.”

-SCA 2.2



SCA Hardware Requirements

- *“Requirements placed on hardware objects by the SCA reflect a balance between the need to support extendibility and interchangeability, and the support of technology growth and domain constraints.” - SCA*

- **Device Profile.**

- Each supplied hardware device **shall** be provided with its associated Domain Profile files....

Verify through inspection that Domain Profile files are provided for each such device. Verify through system/unit testing that files are complete and correct.

- **Hardware Critical Interfaces.**

- Hardware critical interfaces **shall** be defined in Interface Control Documents that are available to other parties without restriction.

Verify through inspection that ICDs exist as required.

SCA Hardware Requirements, cont'd

– Interface Standards.

- Hardware critical interfaces **shall** be in accordance with commercial or government standards..., unless there are program performance requirements that require a non-standard interface.

Verify conformance to commercial/government standards.

- If so required, the non-standard interface **shall** be clearly and openly documented to the extent that interfacing or replacement hardware can be developed by other parties without restriction.

OR.. Verify that the open standard in use is adequately documented.

– Interface Selection.

- In addition to the above, interface selection **should** consider the availability of supporting products that have wide usage, are available from multiple vendors, and are expected to have long-term support in the industry.

Evaluate selection tradeoff for supporting products.

SCA Hardware Requirements, cont'd

– Form Factor.

- The form factor of the hardware objects **should** be, where practical, in accordance with commercial standards.

Evaluate use of commercial standards as required.

– Modularity.

- The partitioning of the hardware architecture into modules **should** be chosen to allow for ease of upgrade through technology insertion or replacement of modules based on form, fit, and function.

– *Evaluate partitioning with consideration for technology insertion.*

JTRS

SCA Issues



SCA Issues

- Service Definitions
- SCA Evolution

Services

- SCA defines Event, File, and Log services
- Additional services are needed for radio operation; for services common to multiple waveforms, APIs and operations should be captured in SCA
 - Time / Location
 - RF & Modem (e.g. Receiver/Exciter, PA, antenna, etc.)
 - Audio / Data Interface (I/O API defined; may be sufficient)
 - Retransmission
 - etc.

Service Definition Sources

- JTRS contractors
- Software Defined Radio Forum
 - this is an industry group "dedicated to supporting the development, deployment, and use of open architectures for advanced wireless systems"
 - members have contributed to the SCA definition
 - expressing interest in defining these types of services
- Object Management Group (OMG)
 - an international consortium that "produces and maintains industry specifications for interoperable enterprise applications"

Evolution of the SCA

- Cluster 1 is the first production program for SCA-compliant systems
 - will address (and specify) architecture aspects not currently complete in SCA
 - realization of Security Supplement / API
 - additional waveform APIs defined and submitted
 - services defined and submitted
 - input from other Cluster customers and contractors will influence updates
- SCA being submitted to OMG as international standard
 - segmented to correspond to their format / approach
 - need to achieve consensus from a larger body of interested parties
 - resulting standard will differ from JPO-controlled SCA update activities

Evolution of the SCA, cont'd

- JTRS community needs to decide the future course now
 - actively working with OMG and adopting the resulting standard for JTRS procurements will require software updates to Cluster 1, but should also result in more COTS availability (applications and devices)
 - portions of the SCA have been introduced and are being considered for acceptance
 - roadmap developed for the remainder of the SCA
 - keeping tighter control on SCA changes will reduce software growth costs, with a standard applicable to military communications systems



JTRS SCA Summary

- The SCA provides the framework architecture to achieve the JTRS goals
- Implementation is underway
 - lessons learned to date support the approach and viability of the SCA
- Continued open dialog, support and interaction of all JTRS-interested parties is essential for success

Open Discussion