Accelerating SCA Compliance Testing with Advanced Development Tools

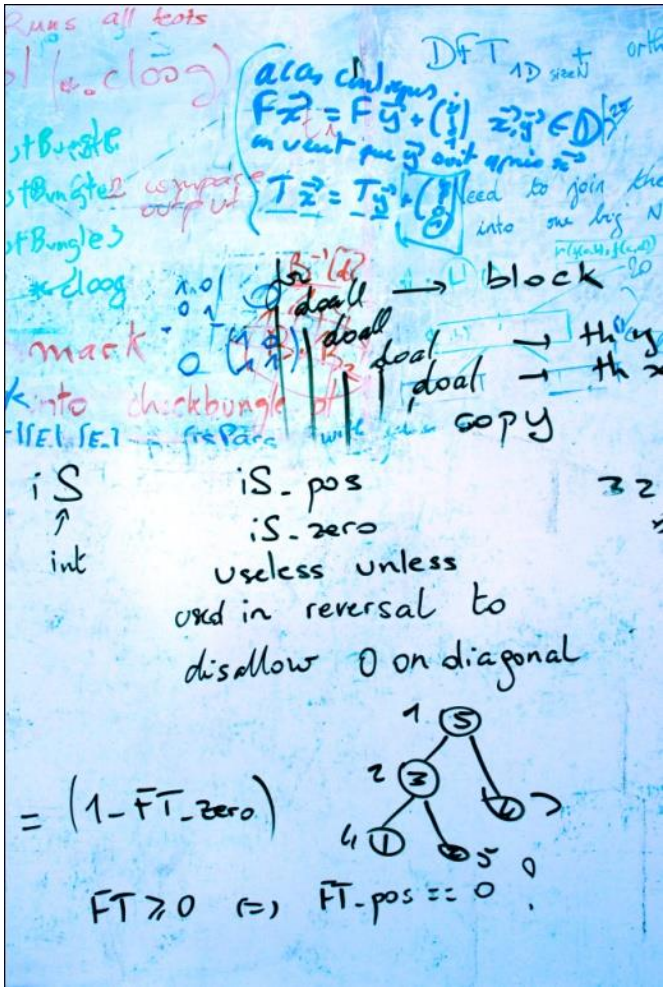
Jonathan Springer *
Steve Bernier ‡
James Ezick *
Juan Pablo Zamora Zapata ‡

* Reservoir Labs, Inc.
New York, NY

‡ NordiaSoft
Gatineau, Québec

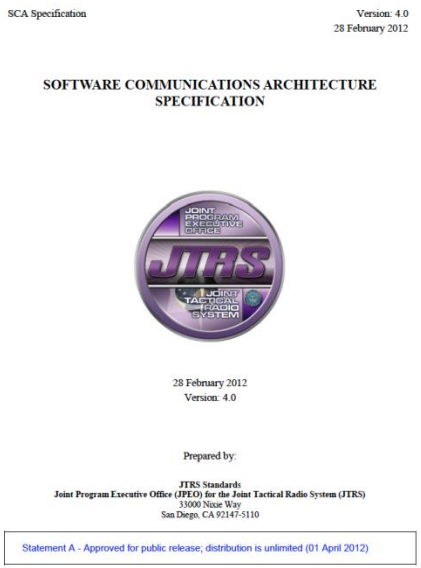
WInnComm 2015
WInnF Conference on
Wireless Communications
Technologies and
Software Defined Radio

25 March 2015



Holy Grail

The dream …


SCA Specification

Version: 4.0
28 February 2012

SOFTWARE COMMUNICATIONS ARCHITECTURE SPECIFICATION

28 February 2012
Version: 4.0

Prepared by:

JTRS Standards
Joint Program Executive Office (JPEO) for the Joint Tactical Radio System (JTRS)
33000 Nixie Way
San Diego, CA 92147-5110

Statement A - Approved for public release; distribution is unlimited (01 April 2012)

Automatic Process

Plain Text Specification Document

Fast Automated Test
Pass: Compliant!
Fail: And here's why …

The reality …

Testing is a time-intensive, expensive, imprecise process that requires a combination of tools – tools that require significant domain expertise to construct and certify

Testing Reality

Specification checking is actually complex

- Lots of individual requirements
  - Over 100 AP, over 500 OE
- Many different kinds of requirements
  - Some statically testable, some dynamically testable
  - Some cannot be perfectly tested (even in theory)
  - Some cannot be automatically tested (e. g. documentation)

Yet testing must be done for specification to be meaningful

- Free riders erode confidence in the specification

Model Based Development Environment (MBDE)

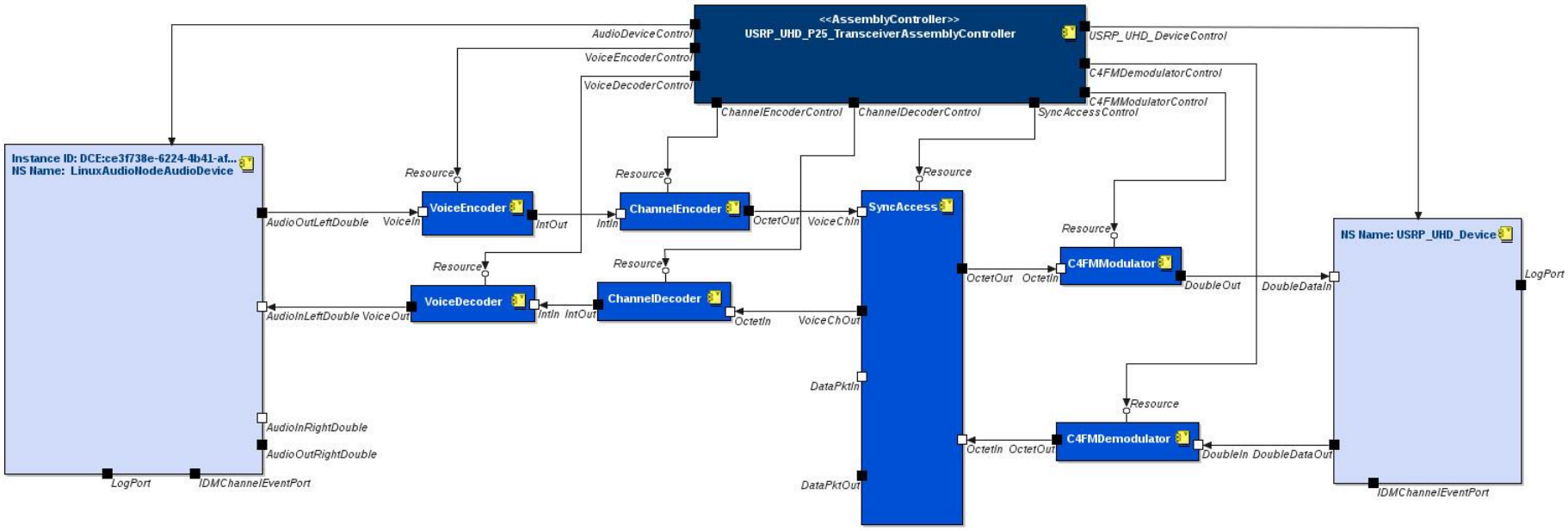
SCA's advantage: Component-Based Development

- Shifts emphasis from *programming software* to *composing software systems*
- Units of composition are *components*

A *Model-Based Development Environment* (MBDE) extends this abstraction to the development process.

- Compose systems from high-level descriptions.
  - This description is the *model*
- Tool generates code artifacts automatically from the model

MBDE Graphical Model View



Model view from NordiaSoft SCA Architect

Model view shows components and how they connect

- Developer edits the model directly
- Code is generated automatically from the model
  - "Business logic" still has to be written manually

MBDE vs Custom Component Development

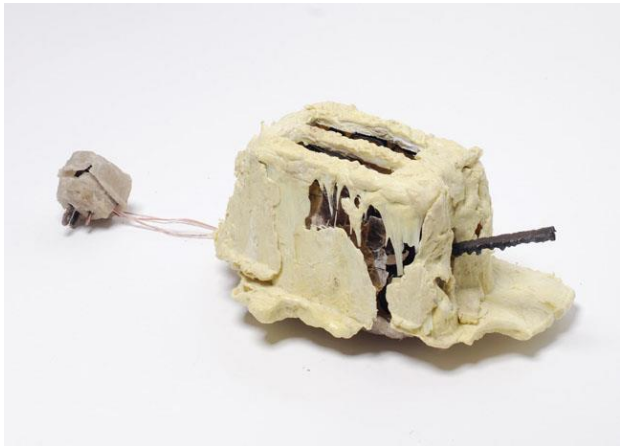
Model Based Development Environment

- Higher-level structure specified in terms of reusable building blocks
- Development environment generates all code to realize the high-level structure
- Enables development leverage such as GUI drag-and-drop

Automatically Generated

Custom Component Development

- Developer must create components from scratch
- Requires intimate knowledge of the specification
- High-level view not available



Generated from Scratch

The Toaster Project, Thomas Thwaites, 2011

Compliance Testing with MBDE

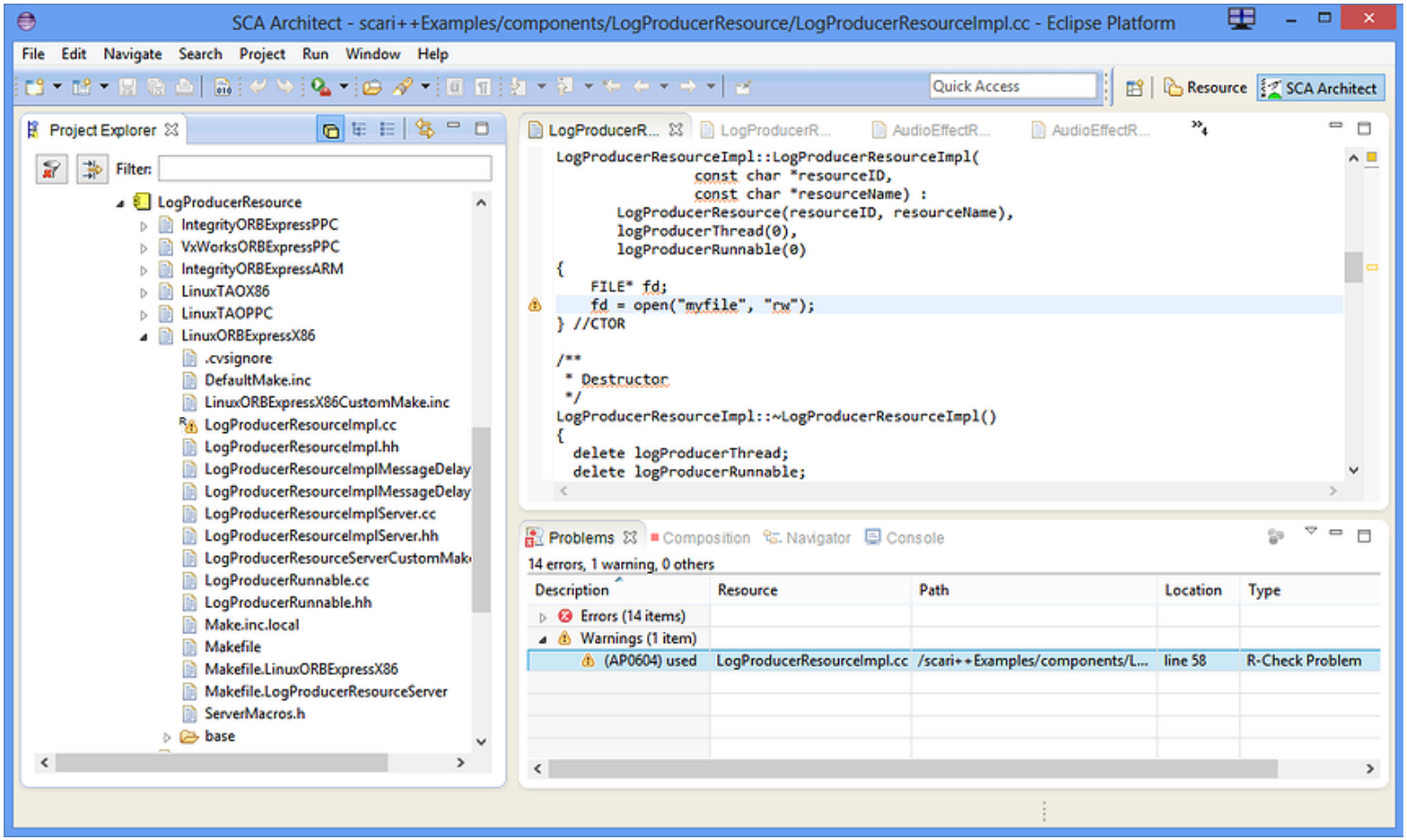
Use of an MBDE simplifies testing

- Reduces custom coding to the business logic
  - Model-generated code is *correct by construction*
- Integrates with static specification checkers
  - Facilitates testing during development
- Supports automated test generation

GateHouse and Inmarsat used MBDE SCA Architect to create the first SCA-compliant commercial satellite communications waveform

> *"We had a very successful experience with JTEL, the entire SCA and API implementation verification took only 4 days. Generating SCA-compliant code from models really works!"*
>
> *—PM Claus Krohn Vesterholt*

MBDE with Integrated Static Specification Checking



SCA Architect with R-Check SCA plugin

Impact of MBDE on Test Lab

The Test Lab is responsible for certifying that a vendor software meets the specification

- Doesn't use MBDE itself, but benefits from vendors' use

Three ways Test Lab can take advantage of this:

- **Pre-Testing**: Vendor is able to test against the specification prior to sending to Test Lab for certification
- **Pre-Certification**: Vendor uses Test Lab accredited tools and submits tool output as evidence for certification
- **Self-Certification**: Vendor uses Test Lab accredited tools and publishes tool output along with software itself

Pre-Testing is current state of the art

- Pre- and Self-Certification are research ideas

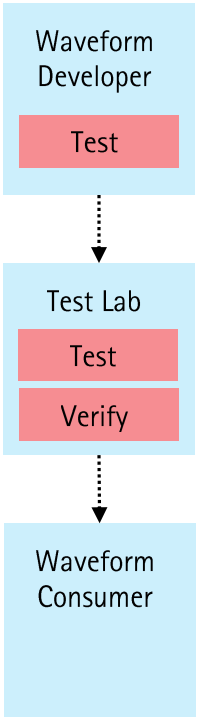
Pre-Testing and the Test Lab

MBDE accelerates Test Lab pre-testing

- Automatically generated code should be correct by construction
- MBDE supports developer in static testing before source code reaches the test lab, reducing errors
- Visual layout provided by MBDE organizes test plan.
- MBDE structure facilitates dynamic tests

Impacts

- Reduce workload for the Test Lab
- Shorten test cycle time



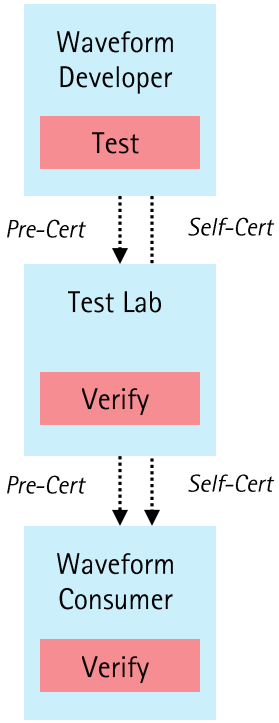
Pre-Certification and Self-Certification

Pre- and Self-Certification offload certification

- Have vendors run the tool instead of the Test Lab

Basic idea: Certification tool generates a checksum

- Certification tool emits a checksum with its results
- Checksum is a function of
  - Test results
  - Source code
  - Test configuration (run options & configuration files)
- Checksum can be independently verified
  - Pre-Certification: checksum verified by Test Lab
  - Self-Certification: checksum verified by any partner



Making Testing More Flexible: Pitchfork

Widespread deployment of automated specification checking tool creates a bottleneck

- Specifications themselves need to be coded by specification checking tool developer
- Test Labs would like to write their own automated checks.
- Software vendors would like to write additional non-specification checks
- New specification checks should be distributable without requiring new version of the tool

Solution: create a specification language understood by the tool

- We call it "Pitchfork"

Pitchfork Use Cases

Finding and capturing known code defects & safety issues

- Addresses "not worth fixing" problem by detecting weaknesses as code is being developed – get critical weaknesses out of code from Day 1
- Adds supplemental support to assist in prioritizing weaknesses and mapping to vulnerabilities

API testing applied to source code

- Proper, consistent, sane implementation in component context
- Providing automated IDL → Pitchfork translation as a starting point

Portability conventions and profiling

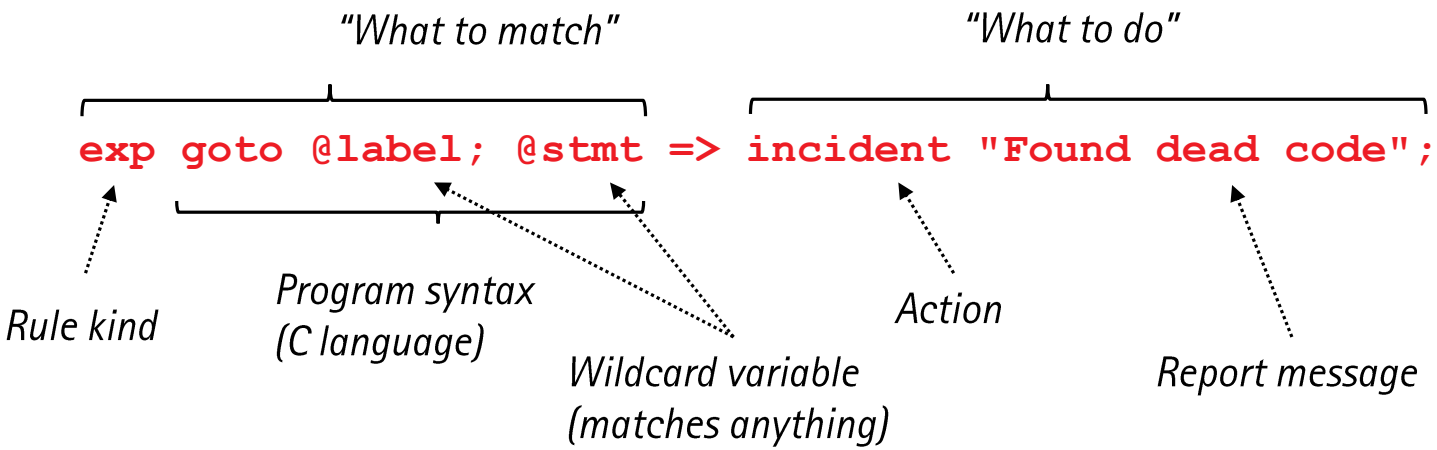
- Capture best practices
- Generate reports that profile API usage for compatibility comparison

Advanced AEP testing

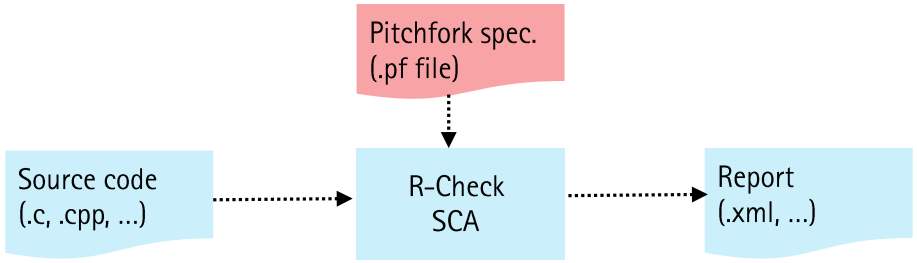
- Example: Sequences of pthread_* calls

Pitchfork in R-Check SCA

Concept: Add analysis capability to R-Check SCA as rules that map sequences of code patterns to actions



R-Check SCA Integration



Pitchfork Example: Taint Tracking

Use of tainted input in file I/O

```
exp    scanf(@buf, @str)
    @@ strcat(@filename, @buf)
    @@ open(@filename)
  => incident "User input in filename";
```

Using program input for filename may allow attacker to access unintended files.

- Rule looks for a particular *sequence* of code
- Statements are related through shared use of metavariables
  - `@buf`, `@filename`
- Incident action reports when the pattern is matched

Pitchfork Example: API Implementation

Ensure code has desired structure

```
exc Audible::InvalidToneId(
                msg : string)
  => api "Audible API";

str Audible::SimpleToneProfile(
                frequencyInHz        : uint4,
                durationPerBurstInMs : uint4,
                repeatIntervalInMs   : uint4)
  => api "Audible API";

fun Audible::createTone(
                SimpleToneProfile)
    : uint4 raises InvalidToneProfile
  => api "Audible API";
```

Abbreviated from *AudibleAlertsAndAlarms*

- API action reports when the pattern is ***not*** found
- Supports testing of proprietary APIs

Summary

Reaping benefits of SCA depends on efficient development of SCA-compliant applications and core frameworks

- Reduced risk, cost, and time-to-market
- Protects against free riders in the SCA ecosystem

Model-based development environments facilitate more than just the development process:

- Facilitates compliance
- Facilitates compliance checking

The increasing level of abstraction enabled by component-based development facilitates configurable specifications

- Specification languages such as Pitchfork become practical
- Useful for checking beyond the specification itself

Acknowledgements

R-Check SCA development funded by SPAWAR under Navy Phase II.5 SBIR Contract
"Static Analysis Tool for Interface Compliance Verification and Program Comprehension"

Thank You

For more information on R-Check SCA

- https://www.reservoir.com/rchecksca

For more information on SCA Architect

- http://www.NordiaSoft.com/

Contact us by email

- rcheck-support@reservoir.com
- info@NordiaSoft.com

Visit our booths!

Reservoir Labs