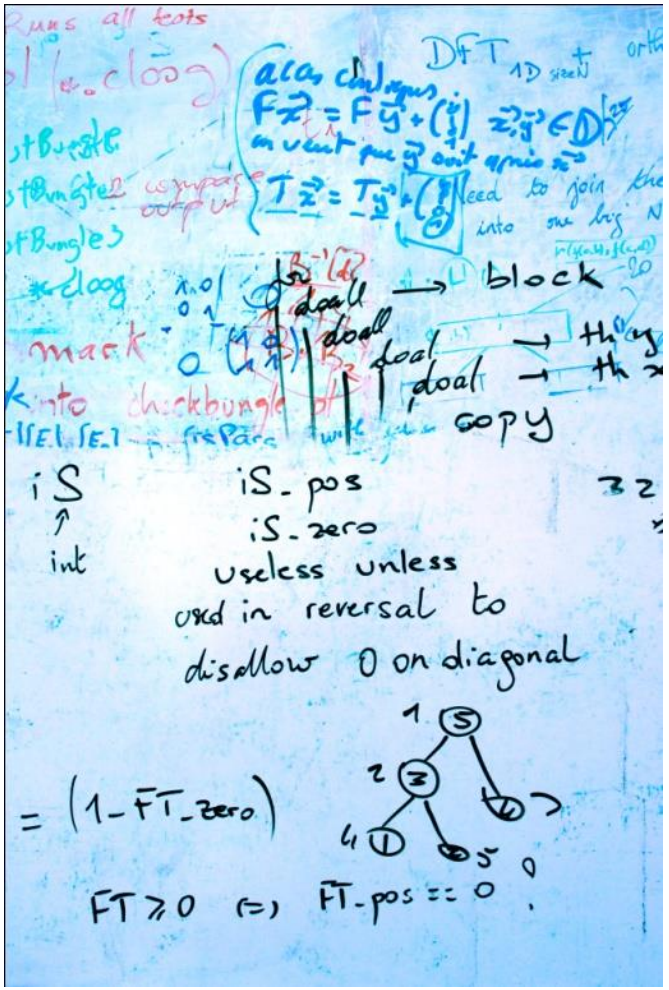
The Continued Evolution of R-Check® SCA

SCA 4.1 Tools Workshop

James Ezick
R-Check SCA Project Manager

Reservoir Labs, Inc.
New York, NY



WInnComm '16
Wireless Innovation Forum
Conference on
Wireless Communications Technologies
and Software Defined Radio

16 March 2016

About the R-Check SCA Technology (1/2)

R-Check SCA is a static analysis tool for the SCA

- Focus is SCA compliance test
- Parses and analyses code (does not require build or platform)
- C/C++: Compiler grade front-end
  - Full preprocessor emulation
  - Full access to syntactic type and context information
  - Relaxed parsing to support missing library files, header files, #define options
- CORBA IDL
- SCA XML Domain Profile files
- Support cross-file analysis, consistency testing across entire SCA project
  - Includes cross-file call graphs for analysis of exceptions
- Includes both GUI and equivalent command-line functionality

Static Analysis refers to a mode of analysis that seeks to find defects through inspection of source code rather than through the execution of the program

- Analyzes all possible program paths without bias
- Can be run at any point the in development cycle
- Integrates with source code development environments

About the R-Check SCA Technology (2/2)

Performance mirrors compilation time

- Waveforms: ~minutes using desktop hardware
- Support parallel processing of source files
- Scales to thousands of files, millions of lines of code – analyzed "as is"
- Support for both Microsoft Windows® and Linux™ Platforms

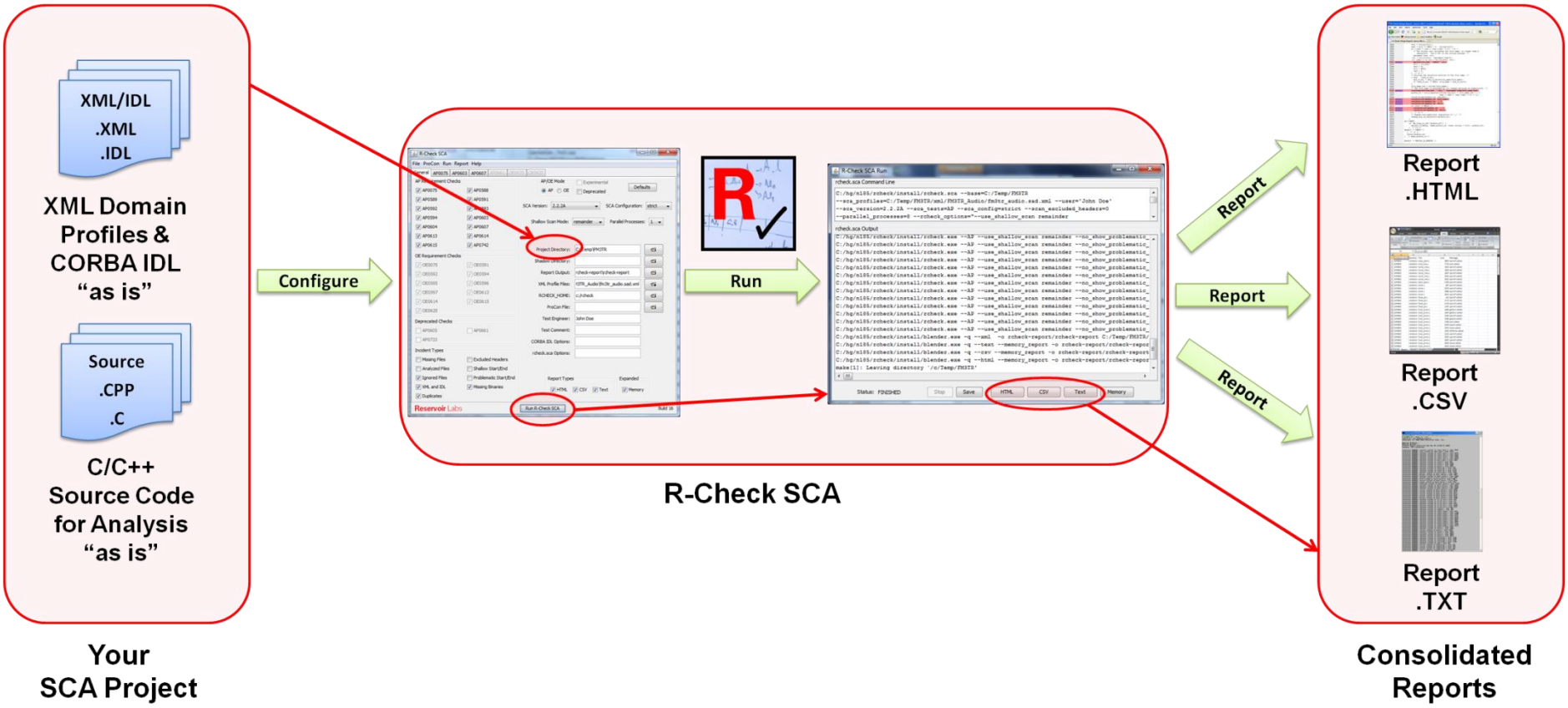
Produces integrated reports for SCA 2.2 and 2.2.2 and includes preview support for SCA 4.0

- Support HTML, CSV (Excel) and text generated from a single XML schema
- Memory report – assist with finding memory leaks
- Pitchfork – pattern-based rule language for API checking and custom rules
- Code coverage report –
  ensure 100% coverage (even code excluded by preprocessor)

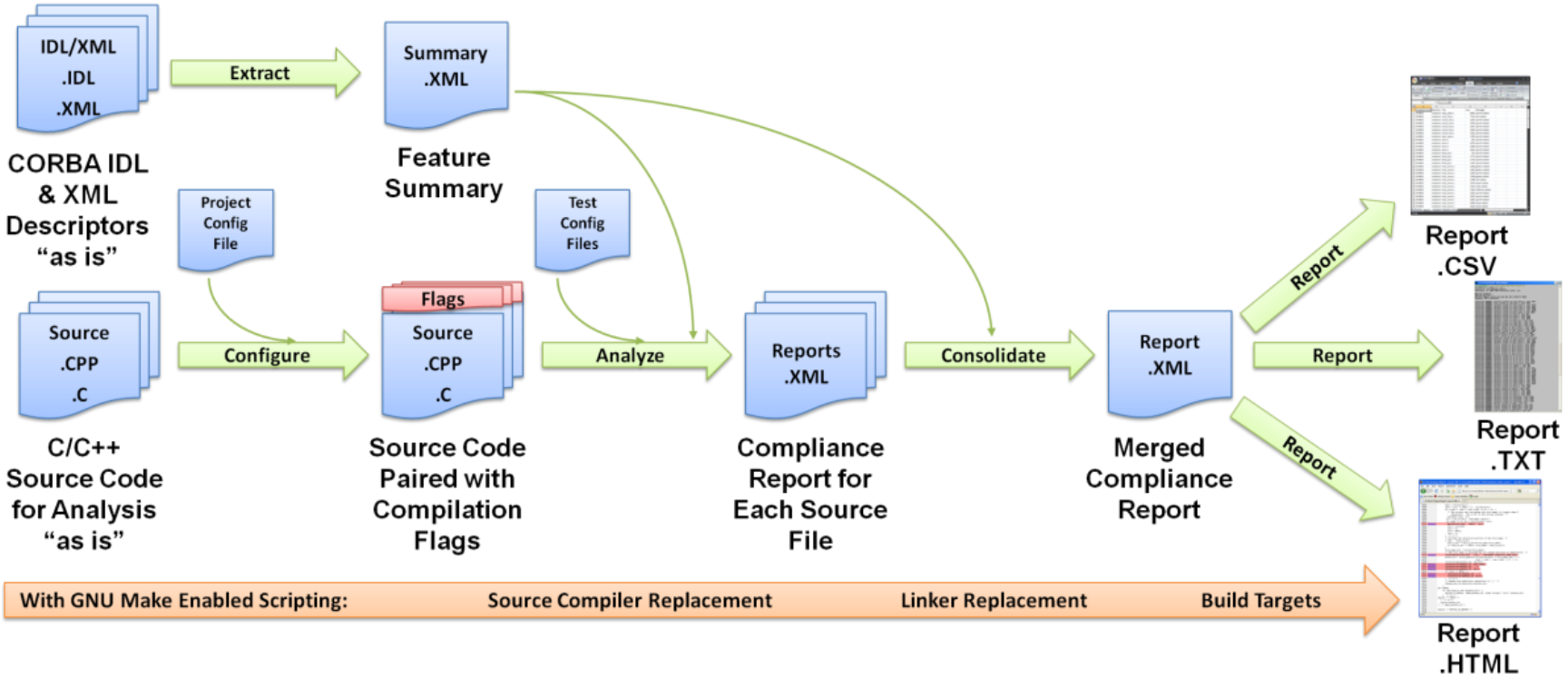
Development significantly supported by US Government Small Business Innovation Research (SBIR) investment

- Since 2008
- Practical engineering while fostering technology innovation

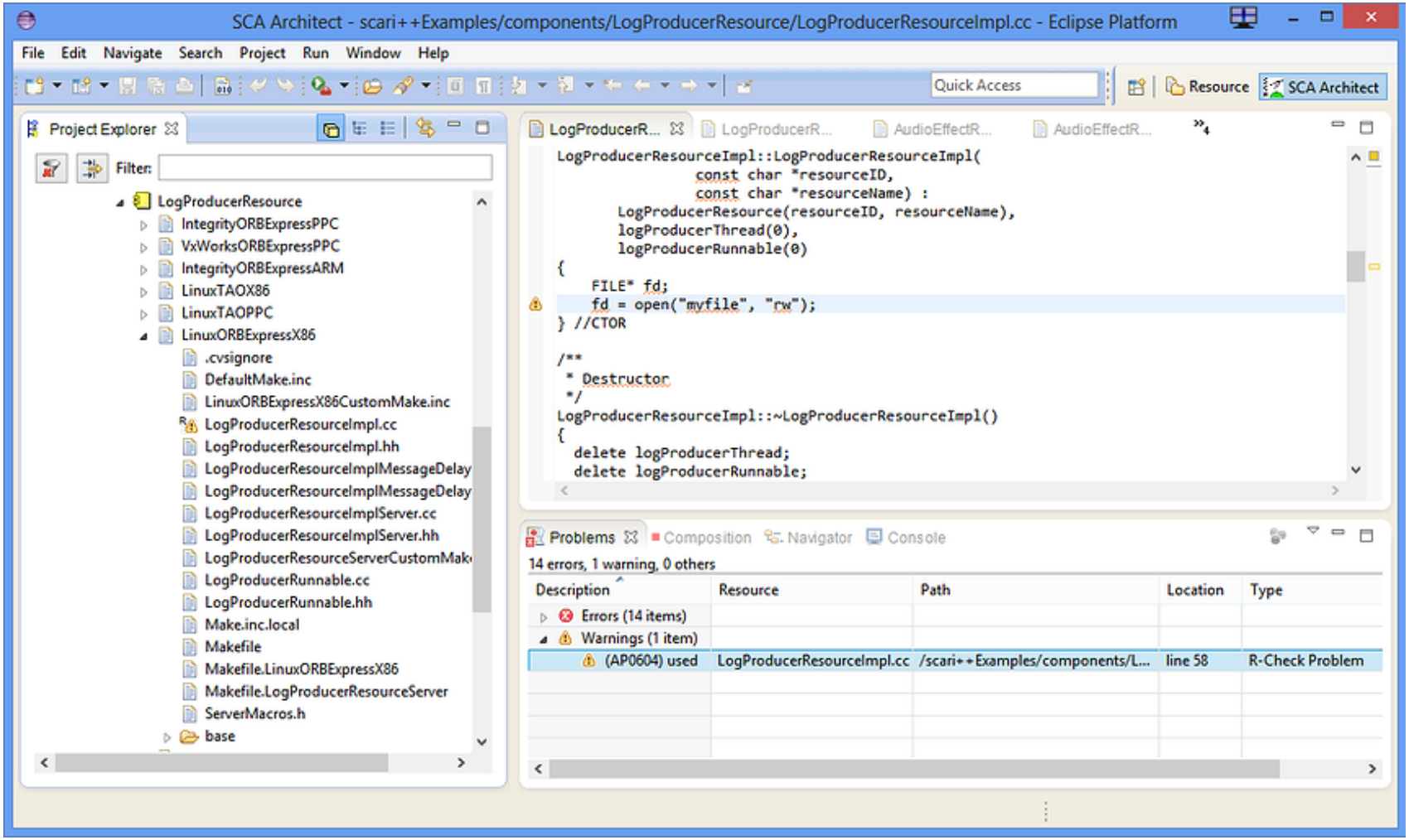
R-Check SCA: Basic Workflow



R-Check SCA: Detailed Workflow



R-Check SCA Adapter Plugin for NordiaSoft's SCA Architect



SCA Compliance Testing

Goals

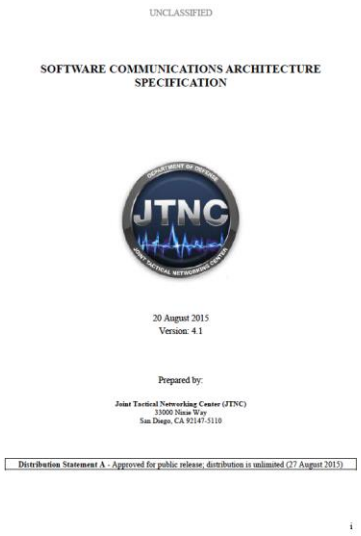
- Reduce test and certification time for both Applications and Operating Environments
  - JTEL using 1.17.5: up to 3.5 weeks saved for an AP, up to 7 weeks saved for an OE
- Improve test accuracy and transparency
- Increase flexibility in the development of new tests
- Reduce difficulty in adopting the SCA
- Simplify the production of repeatable, reproducible test results

Version	Additional Coverage Highlights	SCA 2.2/2.2.2 Requirements	SCA 4.0 Requirements
R-Check SCA 1.17.5 Jul. 2013	Core POSIX (AEP), Minimum CORBA, File Interfaces, Memory reclamation through releaseObject(), XML, IDL requirements	25	0
R-Check SCA 2015 Q2'2015	Component lifecycle, Additional XML requirements, "Shall raise exception ..." SCA 4.0 support for equivalent SCA 2.2.2 tests, SCA 4.0 "shall realize interface", SCA 4.0 "shall accept executable parameters"	153	134

R-Check SCA Requirement Touch Points

Toward SCA 4.1: Holy Grail of Software Testing

The dream …


UNCLASSIFIED

SOFTWARE COMMUNICATIONS ARCHITECTURE SPECIFICATION

JTNC

20 August 2015
Version: 4.1

Prepared by:

Joint Tactical Networking Center (JTNC)
33000 Nixie Way
San Diego, CA 92147-5110

Distribution Statement A - Approved for public release; distribution is unlimited (27 August 2015)

Automatic Process

Plain Text Specification Document

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

The reality …

Testing remains a challenging process that will continue to require and benefit from a combination of tools and processes

Goals for Supporting SCA 4.1

Migrate existing SCA 4.0 support to 4.1

- Completed requirement-by-requirement review

Support C++11

- Upgrades to C++ parsing engine

Improve custom API testing with Pitchfork

- Expand Pitchfork to support new types of testing

Maintain IDE support for business logic

- Interactive capability – catch defects at the earliest possible point in the development process

R-Check SCA 2017

Touch point for every SCA 4.1 requirement

- Blackspot: Provide inspection and reference support, even for requirements we cannot fully test

Embrace automatic code generation

- Provide inspection and reference support ever for requirements we cannot fully test

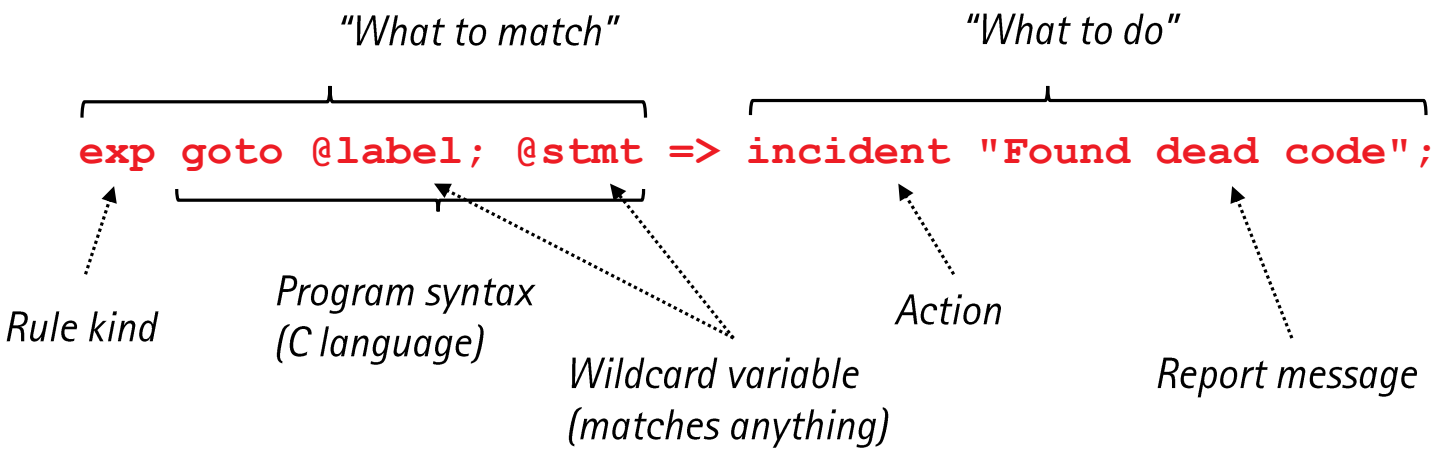
Future Roadmap

Migrate Existing SCA 4.0 Support to SCA 4.1

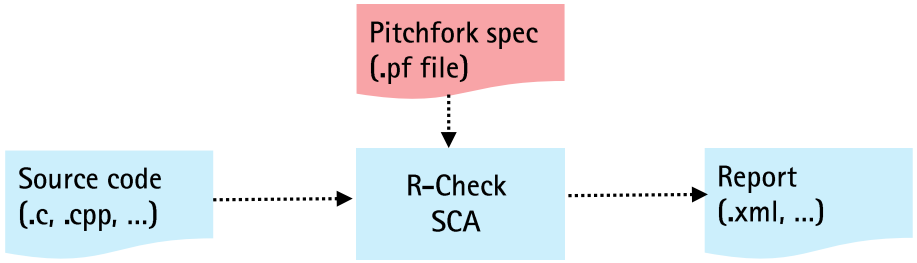
R-Check GUI Category	Rqmt ID	Doc	AP, OE	SCA 4.0 Requirement Text	SCA 4.1 Final Requirement Text	Text Differs?	R-Check implementation from 4.0 to 4.1 Chg?	Description [1]
Exceptions	SCA008	base spec	AP, OE	The *connectUsesPorts* operation shall raise the InvalidPort exception when the input portConnections parameter provides an invalid connection for the specified port.	The *connectUsesPorts* operation shall raise the InvalidPort exception when the input portConnections parameter provides an invalid connection for the specified port.	no	no	
Exceptions	SCA012	base spec	AP, OE	The *disconnectPorts* operation shall raise the InvalidPort exception when the input portDisconnections parameter provides an unknown connection to the *PortAccessor* component.	The *disconnectPorts* operation shall raise the InvalidPort exception when the input portDisconnections parameter provides an unknown connection to the *PortAccessor's* component.	yes	no	The change from "PortAccessor" in 4.0 to "PortAccessor's" in 4.1 does not require the R-Check SCA 4.0 implementation to be changed to support SCA 4.1.
Exceptions	SCA014	base spec	AP, OE	The *getProvidesPorts* operation shall raise an InvalidPort exception when the input portConnections parameter requests undefined connection(s).	The *getProvidesPorts* operation shall raise an InvalidPort exception when the input portConnections parameter requests undefined connection(s).	no	no	
Exceptions	SCA015	base spec	AP, OE	The *initialize* operation shall raise an InitializeError exception when an initialization error occurs.	The *initialize* operation shall raise an InitializeError exception when an initialization error occurs.	no	no	
SCA16	SCA016	base spec	AP, OE	The *releaseObject* operation shall release all internal memory allocated by the component during the life of the component.	The *releaseObject* operation shall release all internal memory allocated by the component during the life of the component.	no	no	
Exceptions	SCA018	base spec	AP, OE	The *releaseObject* operation shall raise a ReleaseError exception when a release error occurs.	The *releaseObject* operation shall raise a ReleaseError exception when a release error occurs.	no	no	
Exceptions	SCA023	base spec	AP, OE	The *runTest* operation shall raise the UnknownTest exception when there is no underlying test implementation that is associated with the input testId given.	The *runTest* operation shall raise the UnknownTest exception when there is no underlying test implementation that is associated with the input testId given.	no	no	
Exceptions	SCA024	base spec	AP, OE	The runTest operation shall raise the CF UnknownProperties exception when the input parameter testValues contains any CF DataTypes that are not known by the component's test implementation or any values that are out of range for the requested test.	The *runTest* operation shall raise the CF::UnknownProperties exception when the input parameter testValues contains any CF::DataTypes that are not known by the component's test implementation or any values that are out of range for the requested test.	yes	no	The change from "CF DataTypes" in 4.0 to "CF::DataTypes" in 4.1 does not require the R-Check SCA 4.0 implementation to be changed to support SCA 4.1.

Pitchfork in R-Check SCA

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



R-Check SCA workflow integration



SCA 4.1 Compliance Requirements

SCA 4.1 Compliance Requirements

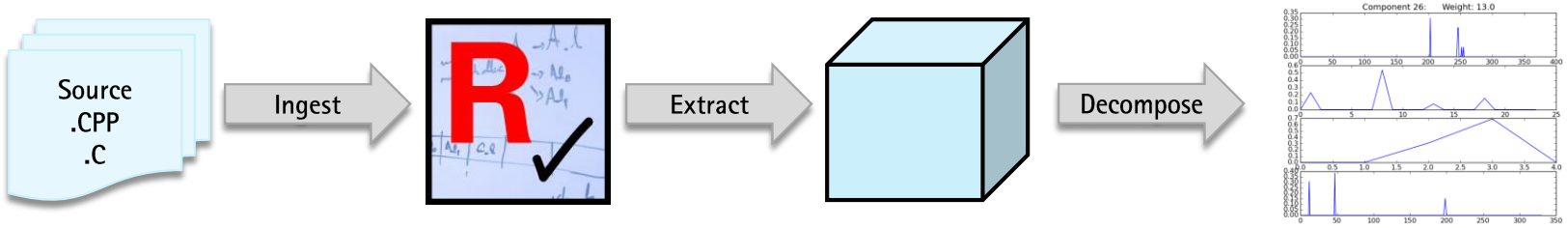
"The X operation shall Y if/when Z occurs."

SCA15: The initialize operation shall raise an InitializeError exception when an initialization error occurs.

SCA16: The releaseObject operation shall release all internal memory allocated by the component during the life of the component.

- Static Analysis
  - Can find *X*, can determine if *Y* and test for *Z* (sometimes) exists
  - Success depends on matching *Y* with *Z* – difficult (undecidable?)
- Dynamic Testing
  - Can run *X*, select a sample of *Z* scenarios, check for *Y*
  - Success depends on coverage of *Z* (usually assume $\neg Z \rightarrow \neg Y$)

Blackspot Workflow



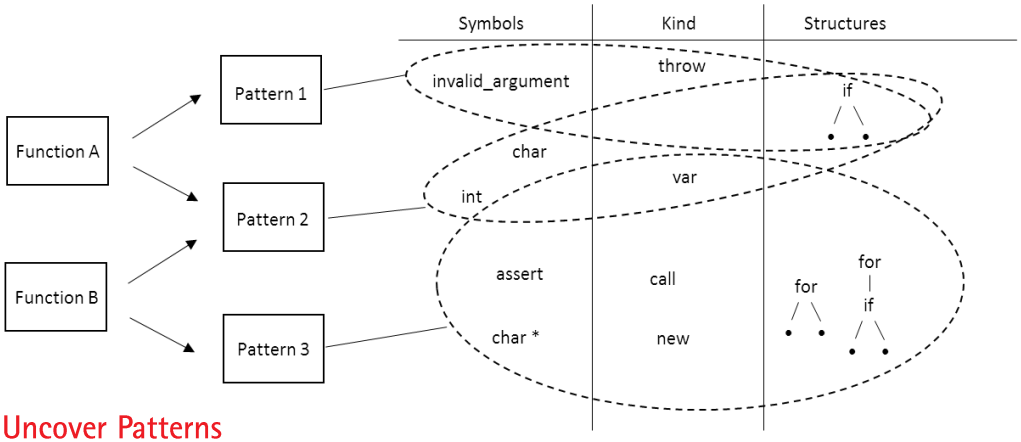
C/C++
Source Code for Analysis "as is"

R-Check SCA
Uses compiler-grade whole project parsing combined with Pitchfork rules to extract markers from code

Pitchfork
Metalanguage that combines syntactic patterns with actions – used to identify AST structures in code

Tensor
Function x *Structure* x *Symbol* x *Kind*

Analyze



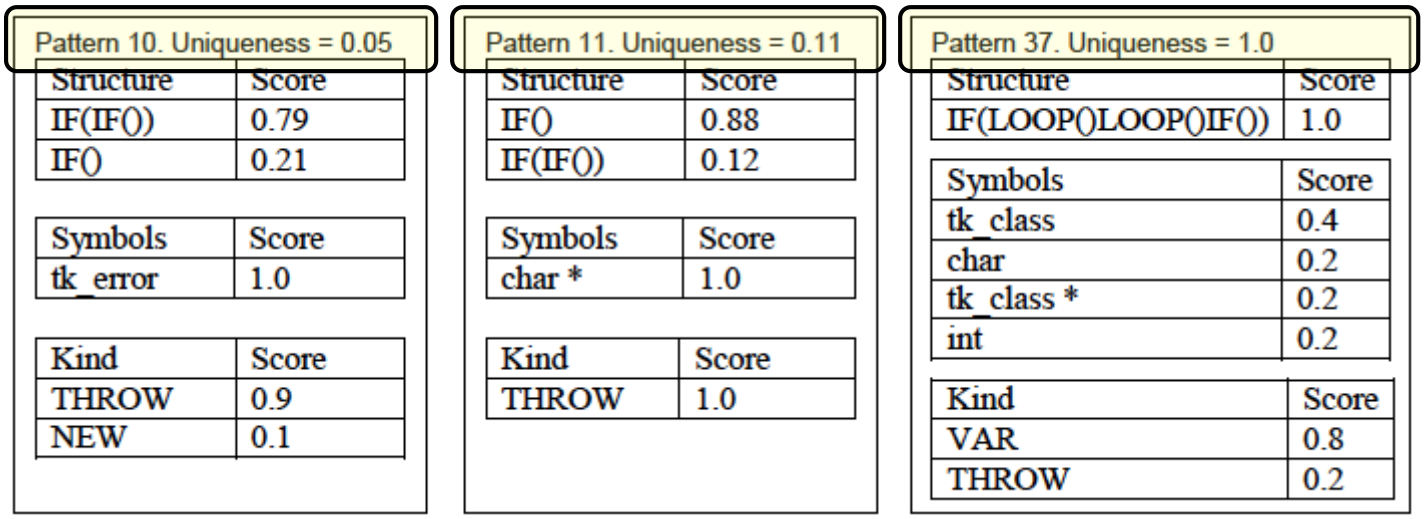
Uncover Patterns

Cluster functions based on similarity of pattern occurrence
Logically organize code for manual inspection

Identify functions that share patterns with known defects
Find "synonyms" of known defects for inspection

Blackspot: Using Tensor Decompositions to Guide Inspection of Source Code
D. Bruns-Smith, J. Ezick, J. McMahon, J. Springer, WInnComm '16.

Getting to a Touch Point for Every SCA Requirement



Pattern 10. Uniqueness = 0.05

Structure	Score
IF(IF())	0.79
IF()	0.21

Symbols	Score
tk_error	1.0

Kind	Score
THROW	0.9
NEW	0.1

Pattern 11. Uniqueness = 0.11

Structure	Score
IF()	0.88
IF(IF())	0.12

Symbols	Score
char *	1.0

Kind	Score
THROW	1.0

Pattern 37. Uniqueness = 1.0

Structure	Score
IF(LOOP()LOOP()IF())	1.0

Symbols	Score
tk_class	0.4
char	0.2
tk_class *	0.2
int	0.2

Kind	Score
VAR	0.8
THROW	0.2

Example: [SCA 2.2.2] AP0090: The getPort operation shall raise an UnknownPort exception if the port name is invalid.

Uniqueness Score captures the number of functions in which a particular code pattern occurs (# functions = 1/uniqueness score)

From the breakdown of patterns –
Code inspection tasks can be organized in a logical way –
Sort relevant functions by pattern – or – sort by pattern across requirements

Supporting Model Based Development

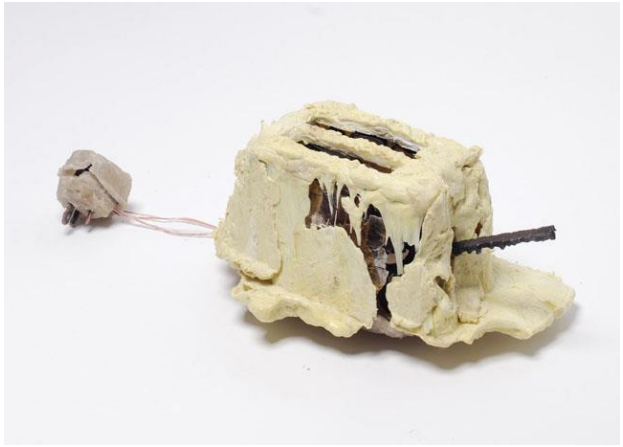
Model Based Development Environment

- 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

Unlocks New Models of Compliance Testing

- **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

Automatically Generated



Generated from Scratch

The Toaster Project, Thomas Thwaites, 2011

Accelerating SCA Compliance Testing with Advanced Development Tools
J. Springer, S. Bernier, J. Ezick and J.P. Zamora Zapata, WInnComm '15.

Thank You

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

For more information on R-Check SCA

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

Contact us by email

- rcheck-support@reservoir.com