Carleton
UNIVERSITY

Reconciling Offshore Outsourcing with Model Based Testing

Dave Arnold, Jean-Pierre Corriveau, and Wei Shi
Carleton University, Ottawa, Canada

June 17th, SEAFOOD 2010

My Background

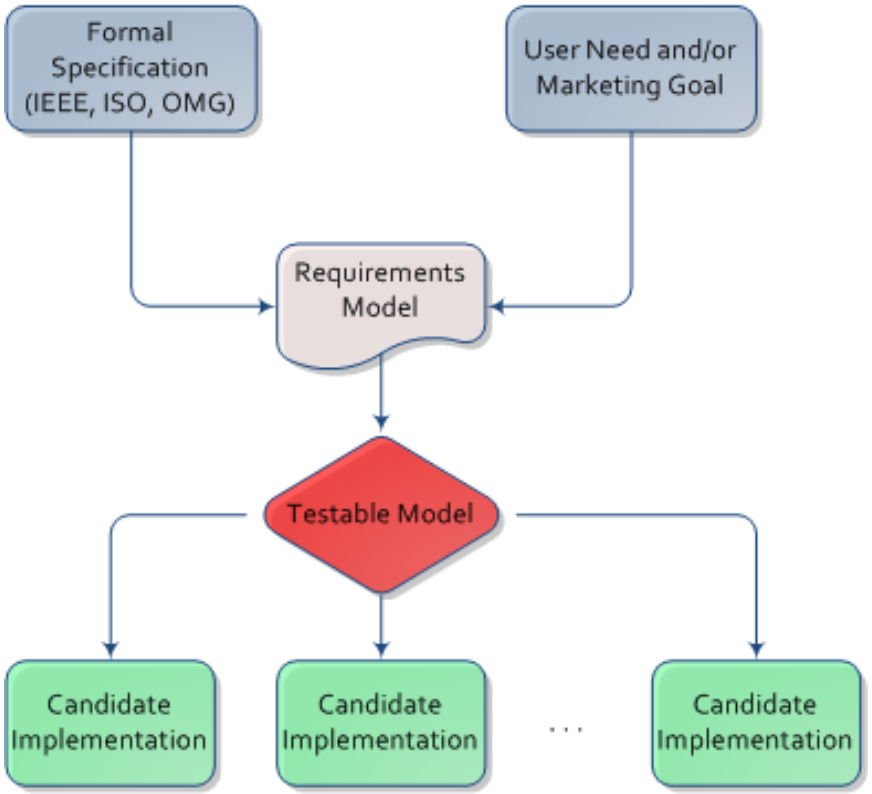
Jean-Pierre Corriveau

- At Nortel: one of the original creators of ObjecTime (ROSE-RT) in 1986
  - Consulting/teaching in telecoms since 1991
- In collaboration with F. Bordeleau (Zeligsoft):
  - Worked with Raytheon on modeling for compliance the software radio specification
- In collaboration with staff at Amazon and Bitheads:
  - Discussing outsourcing and testing in industry

Premises

- Outsourcing is a business relationship:
  - Any business relationship needs some form of *contract:*
    - Must define deliverables and dates
    - Must state *how* quality is verified
- Compliance/conformance testing must be a key facet of an *offshore* outsourcing contract.
  - We require automated validation against an **actual** implementation!

Model-Based Testing



Testable Models

- We require a testable model capable of automatically generating/instrumenting executable checks.

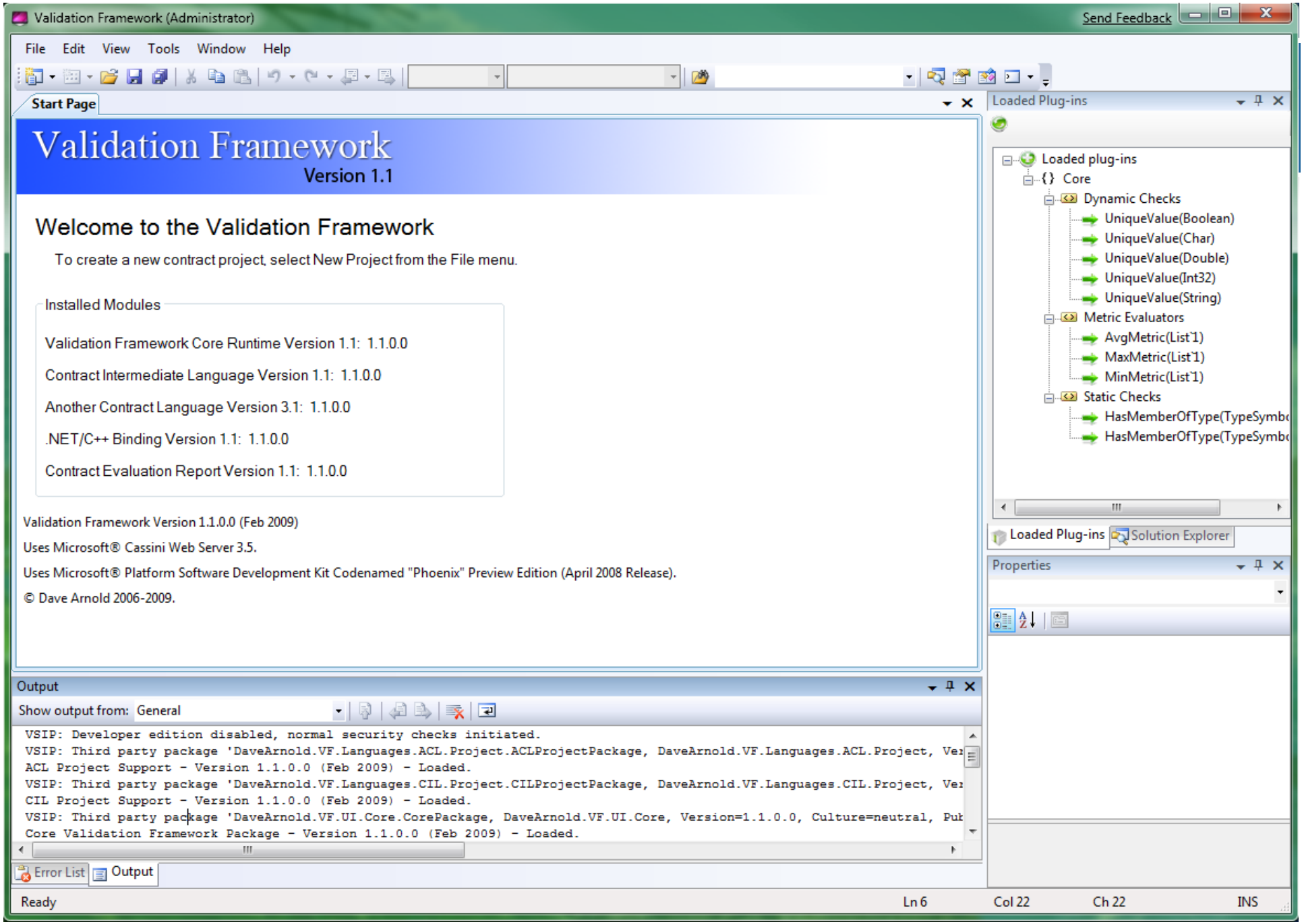
- Such a testable model must support:
  - The capture of functional and non-functional requirements
  - Testability of the requirements model
  - Executability of the generated static and dynamic checks
  - Semantics rooted in the notions of *responsibilities* and *scenarios*
  - Abstraction of the testable model over several possible implementations

- Current approaches to validation typically do not offer a testable requirements model with the above characteristics…

Our Approach

- The challenge with the development of MBT tools lies in the ability to easily express the testable model at a level of abstraction that is implementation independent, yet executable.

- We seek to create an *open* framework for the *specification* and *execution* of a testable model against an implementation:
  - http://vf.davearnold.ca/


Validation Framework (Administrator)
Send Feedback
File Edit View Tools Window Help
Start Page
Validation Framework
Version 1.1
Welcome to the Validation Framework
To create a new contract project, select New Project from the File menu.
Installed Modules
Validation Framework Core Runtime Version 1.1: 1.1.0.0
Contract Intermediate Language Version 1.1: 1.1.0.0
Another Contract Language Version 3.1: 1.1.0.0
.NET/C++ Binding Version 1.1: 1.1.0.0
Contract Evaluation Report Version 1.1: 1.1.0.0
Validation Framework Version 1.1.0.0 (Feb 2009)
Uses Microsoft® Cassini Web Server 3.5.
Uses Microsoft® Platform Software Development Kit Codenamed "Phoenix" Preview Edition (April 2008 Release).
© Dave Arnold 2006-2009.
Loaded Plug-ins
Loaded plug-ins
Core
Dynamic Checks
UniqueValue(Boolean)
UniqueValue(Char)
UniqueValue(Double)
UniqueValue(Int32)
UniqueValue(String)
Metric Evaluators
AvgMetric(List`1)
MaxMetric(List`1)
MinMetric(List`1)
Static Checks
HasMemberOfType(TypeSymbo
HasMemberOfType(TypeSymbo
Loaded Plug-ins
Solution Explorer
Properties
Output
Show output from: General
VSIP: Developer edition disabled, normal security checks initiated.
VSIP: Third party package 'DaveArnold.VF.Languages.ACL.Project.ACLProjectPackage, DaveArnold.VF.Languages.ACL.Project, Ver
ACL Project Support - Version 1.1.0.0 (Feb 2009) - Loaded.
VSIP: Third party package 'DaveArnold.VF.Languages.CIL.Project.CILProjectPackage, DaveArnold.VF.Languages.CIL.Project, Ver
CIL Project Support - Version 1.1.0.0 (Feb 2009) - Loaded.
VSIP: Third party package 'DaveArnold.VF.UI.Core.CorePackage, DaveArnold.VF.UI.Core, Version=1.1.0.0, Culture=neutral, Pub
Core Validation Framework Package - Version 1.1.0.0 (Feb 2009) - Loaded.
Error List
Output
Ready
Ln 6
Col 22
Ch 22
INS

An Example Contract (1)

```
Import Core;

Namespace DaveArnold.Examples.School
{
    MainContract University
    {
        Parameters
        {
            [1-100] Scalar Integer InstanceBind UniversityCourses;
            Scalar Integer MaxCoursesForFTStudents = 4;
            Scalar Integer MaxCoursesForPTStudents = 2;
            Scalar Integer PassRate = 70;
            [1-12] Scalar Integer InstanceBind NumTermsToComplete;
        }

        Observability List tCourse Courses();
        Observability List tStudent Students();
```

An Example Contract (2)

```
Responsibility new() {
    Post(Courses().Length() == 0);
    Post(Students().Length() == 0);
}
Responsibility finalize() {
    Pre(Courses().Length() == 0);
    Pre(Students().Length() == 0);
}

Responsibility tCourse CreateCourse(String name, Integer cap) {
    once Scalar Integer oldSize;
    oldSize = PreSet(Courses().Length());
    Post(value.bindpoint.Name() == name);
    Post(value.bindpoint.CapSize() == cap);
    Post(Courses().Length() == oldSize + 1);
    Post(Courses().Contains(value) == true);
}
```

An Example Contract (3)

```
Responsibility ReportMark (tCourse course, tStudent student, Integer mark) {
     choice(mark) < Parameters.PassRate
       { student.bindpoint.failures = student.bindpoint.failures + 1; }

Responsibility RegisterStudentForCourse(tStudent student, tCourse course);

Responsibility CancelCourse(tCourse course)  {
          Pre(Courses().Contains(course) == true));
          Post(Courses().Contains(course) == false));              }

Responsibility CalculatePassFail()              {
          each(Students())
           choice(iterator.bindpoint.failures) >= 2
                     FailStudent(iterator);
           alternative
                     PassStudent(iterator);              }
```

An Example Contract (4)

```
Scenario Term {
    Trigger(new()),
    (
          CreateCourse()[Parameters.UniversityCourses],
          TermStarted(),
          fire(TermStarted),
          LastDayToDrop(),
          fire(LastDayToDrop),
          TermEnded(),
          fire(TermEnded),
          observe(MarksRecorded)[Parameters.UniversityCourses],
          CalculatePassFail(),
          DestroyCourse()[Parameters.UniversityCourses],
          fire(TermComplete)
    )+,
    Terminate(finalize());
}
```

An Example Contract (5)

```
Exports
{
    Type tCourse conforms Course
    {
          Student::tCourse;
    }
    Type tStudent conforms Student
    {
          Course::tStudent;
    }
}
```

Bindings

- We need to connect the testable requirements model (in ACL) to the Implementation Under Test (IUT)
  - This is accomplished through the notion of *bindings*
  - Bindings are a mapping between an ACL element and a IUT element:
    - Contracts → Types (Classes, Structs)
    - Observabilities → A single method or property
    - Responsibilities → One or more methods

Bindings

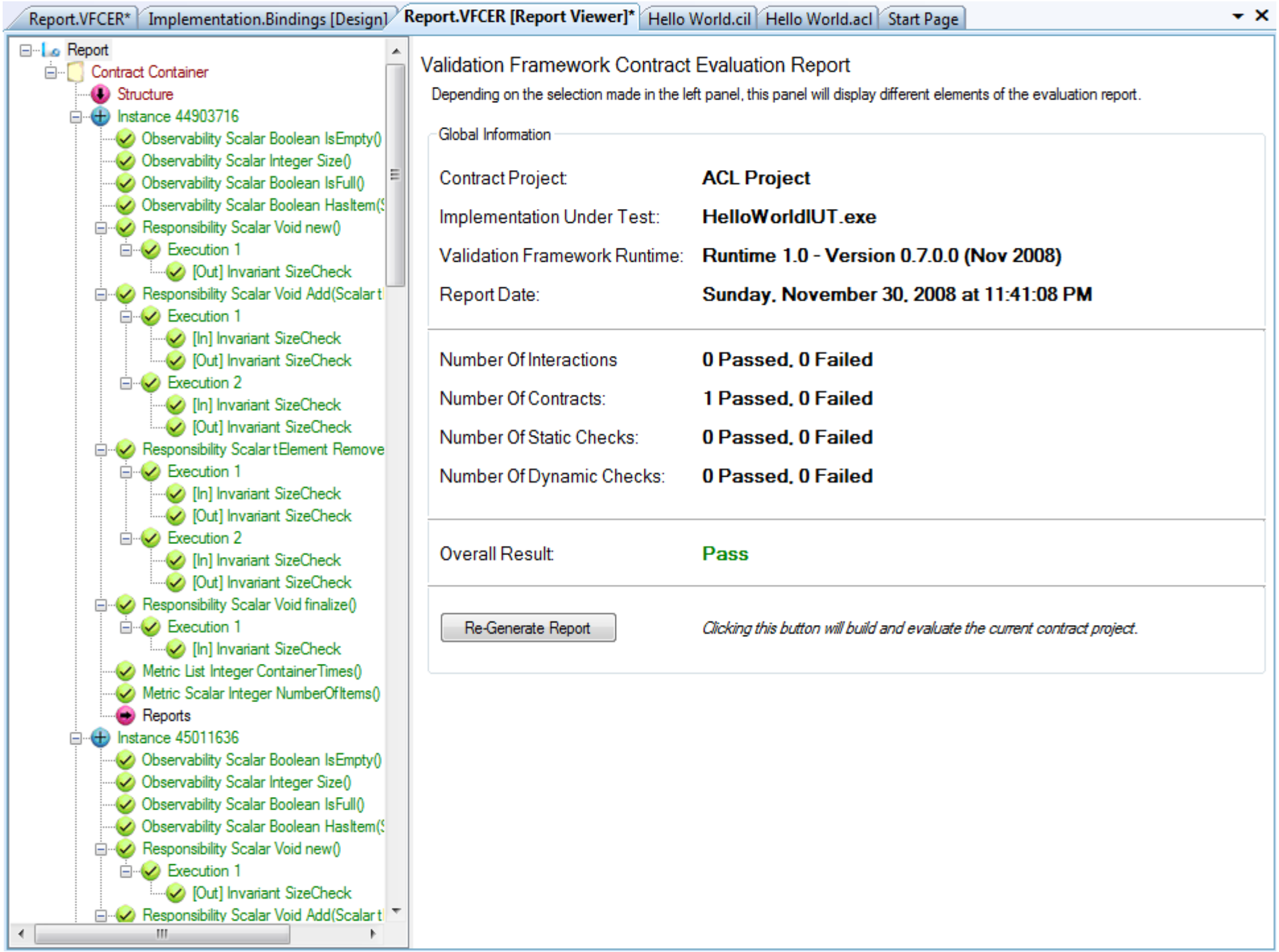
- In order to reduce the dependency on manual binding
  - We use binding extension modules to **infer** as many bindings as possible
    - Modules can be written by third-party developers.
  - When a binding cannot be inferred, a short list of possible bindings is presented, and the user is asked to make a selection

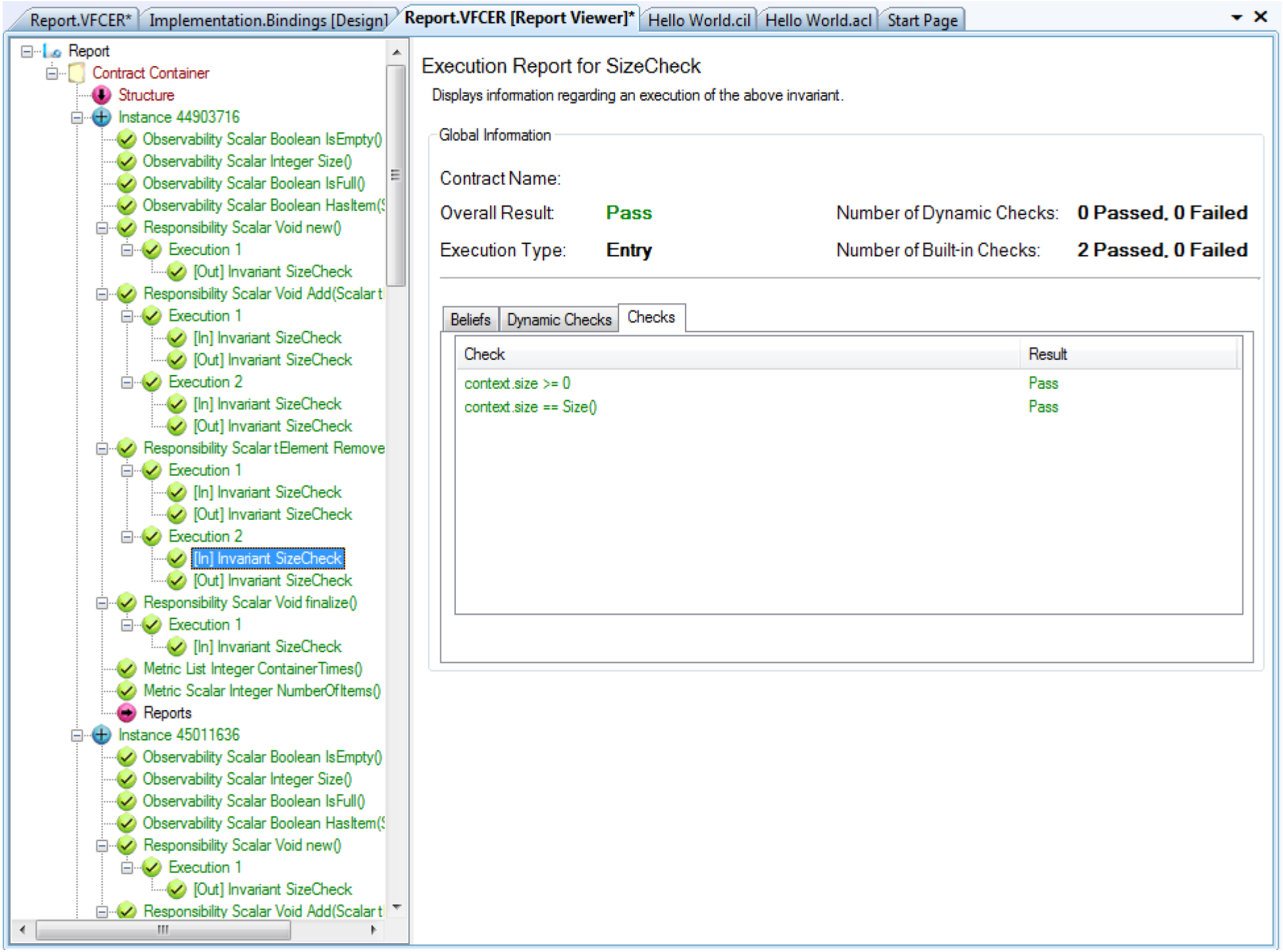
Execution

- The model is then compiled and executed against the IUT
  - Static checks are evaluated
  - The IUT is launched against the runtime
    - Execution is monitored for responsibilities and scenarios
    - Observabilities are invoked as needed by the runtime
    - Metric information is captured (Performance, Security, etc)
  - Metric evaluators determine results based on gathered metric information
- The result is a Contract Evaluation Report (CER)

Contract Evaluation Report

- The CER provides information on the IUT's execution:
  - Static evaluation results
  - For each object instance
    - Information pertaining to any dynamic checks
    - Information regarding the pass/fail of observabilities, responsibilities, and scenarios
      - Preconditions
      - Post-conditions
      - Invariants
      - Beliefs
      - Dynamic Checks
    - The result of metric analysis


Report.VFCER*
Implementation.Bindings [Design]
Report.VFCER [Report Viewer]*
Hello World.cil
Hello World.acl
Start Page
Report
Contract Container
Structure
Instance 44903716
Observability Scalar Boolean IsEmpty()
Observability Scalar Integer Size()
Observability Scalar Boolean IsFull()
Observability Scalar Boolean HasItem(
Responsibility Scalar Void new()
Execution 1
[Out] Invariant SizeCheck
Responsibility Scalar Void Add(Scalar t
Execution 1
[In] Invariant SizeCheck
[Out] Invariant SizeCheck
Execution 2
[In] Invariant SizeCheck
[Out] Invariant SizeCheck
Responsibility Scalar tElement Remove
Execution 1
[In] Invariant SizeCheck
[Out] Invariant SizeCheck
Execution 2
[In] Invariant SizeCheck
[Out] Invariant SizeCheck
Responsibility Scalar Void finalize()
Execution 1
[In] Invariant SizeCheck
Metric List Integer ContainerTimes()
Metric Scalar Integer NumberOfItems()
Reports
Instance 45011636
Observability Scalar Boolean IsEmpty()
Observability Scalar Integer Size()
Observability Scalar Boolean IsFull()
Observability Scalar Boolean HasItem(
Responsibility Scalar Void new()
Execution 1
[Out] Invariant SizeCheck
Responsibility Scalar Void Add(Scalar t
Validation Framework Contract Evaluation Report
Depending on the selection made in the left panel, this panel will display different elements of the evaluation report.
Global Information
Contract Project:
ACL Project
Implementation Under Test::
HelloWorldIUT.exe
Validation Framework Runtime:
Runtime 1.0 - Version 0.7.0.0 (Nov 2008)
Report Date:
Sunday, November 30, 2008 at 11:41:08 PM
Number Of Interactions
0 Passed, 0 Failed
Number Of Contracts:
1 Passed, 0 Failed
Number Of Static Checks:
0 Passed, 0 Failed
Number Of Dynamic Checks:
0 Passed, 0 Failed
Overall Result:
Pass
Re-Generate Report
Clicking this button will build and evaluate the current contract project.


Report.VFCER*
Implementation.Bindings [Design]
Report.VFCER [Report Viewer]*
Hello World.cil
Hello World.acl
Start Page
Report
Contract Container
Structure
Instance 44903716
Observability Scalar Boolean IsEmpty()
Observability Scalar Integer Size()
Observability Scalar Boolean IsFull()
Observability Scalar Boolean HasItem(
Responsibility Scalar Void new()
Execution 1
[Out] Invariant SizeCheck
Responsibility Scalar Void Add(Scalar t
Execution 1
[In] Invariant SizeCheck
[Out] Invariant SizeCheck
Execution 2
[In] Invariant SizeCheck
[Out] Invariant SizeCheck
Responsibility Scalar tElement Remove
Execution 1
[In] Invariant SizeCheck
[Out] Invariant SizeCheck
Execution 2
[In] Invariant SizeCheck
[Out] Invariant SizeCheck
Responsibility Scalar Void finalize()
Execution 1
[In] Invariant SizeCheck
Metric List Integer ContainerTimes()
Metric Scalar Integer NumberOfItems()
Reports
Instance 45011636
Observability Scalar Boolean IsEmpty()
Observability Scalar Integer Size()
Observability Scalar Boolean IsFull()
Observability Scalar Boolean HasItem(
Responsibility Scalar Void new()
Execution 1
[Out] Invariant SizeCheck
Responsibility Scalar Void Add(Scalar t
Execution Report for SizeCheck
Displays information regarding an execution of the above invariant.
Global Information
Contract Name:
Overall Result: Pass
Number of Dynamic Checks: 0 Passed, 0 Failed
Execution Type: Entry
Number of Built-in Checks: 2 Passed, 0 Failed
Beliefs
Dynamic Checks
Checks
Check
Result
context.size >= 0
Pass
context.size == Size()
Pass

Additional Features

- The VF is also able to support
  - Contract refinement/inheritance
  - Atomic / parallel scenario blocks
  - Support for execution against web applications

- The VF consists of 1,355 classes totaling over 260,000 lines of C# and C++ source code

Validation of Our Approach

- Validation of our approach included
  - Individual testing of the ACL and CIL compilers
    - 1,516 individual tests performed
  - Five case studies
    - Basic container
    - Advanced container
    - Web login
    - Grocery store
    - University course registration and term operation
  - Use by a group of graduate students
    - Verify existing case studies
    - Develop small to medium size projects (including army code!)

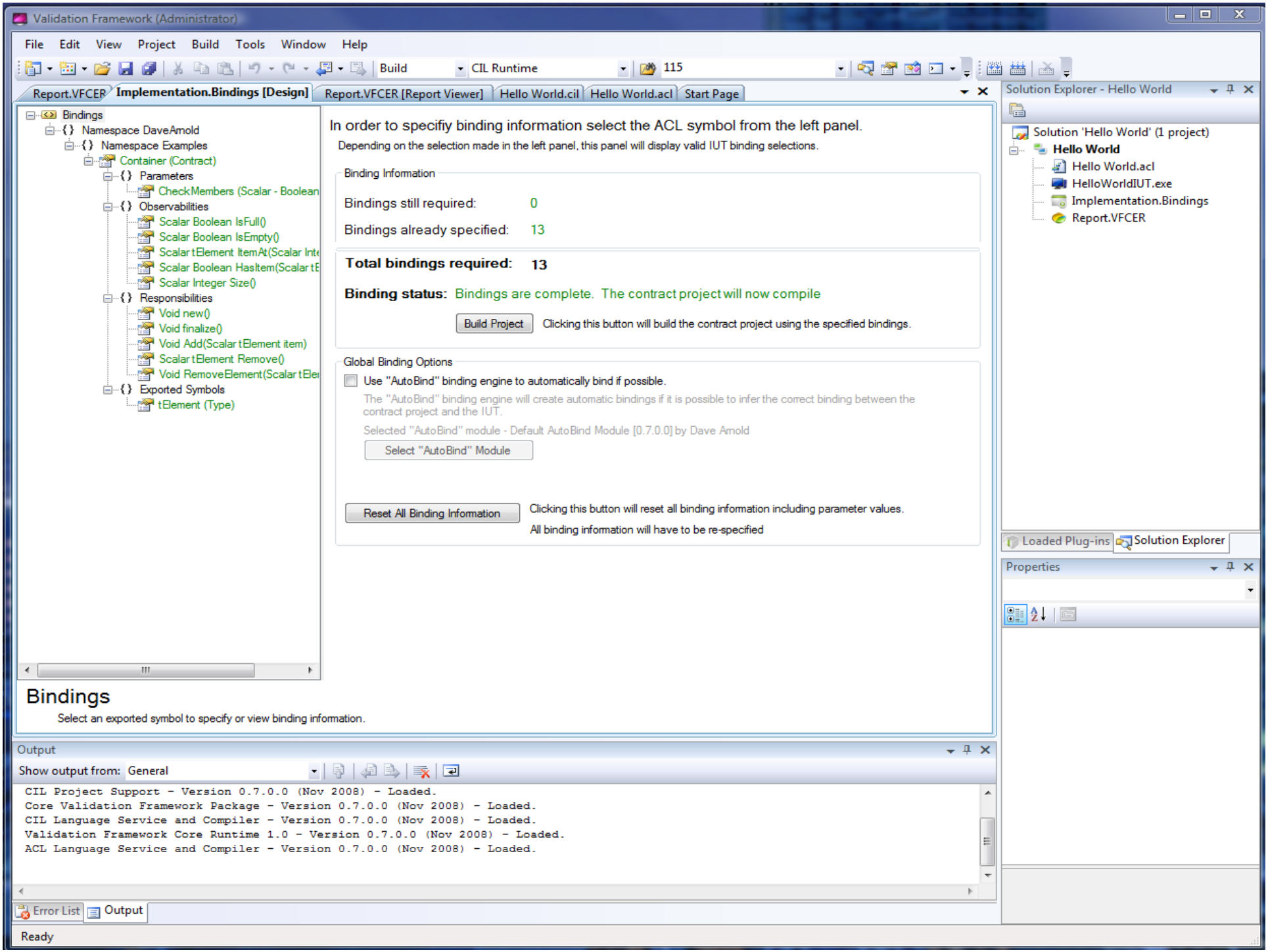
Contributions

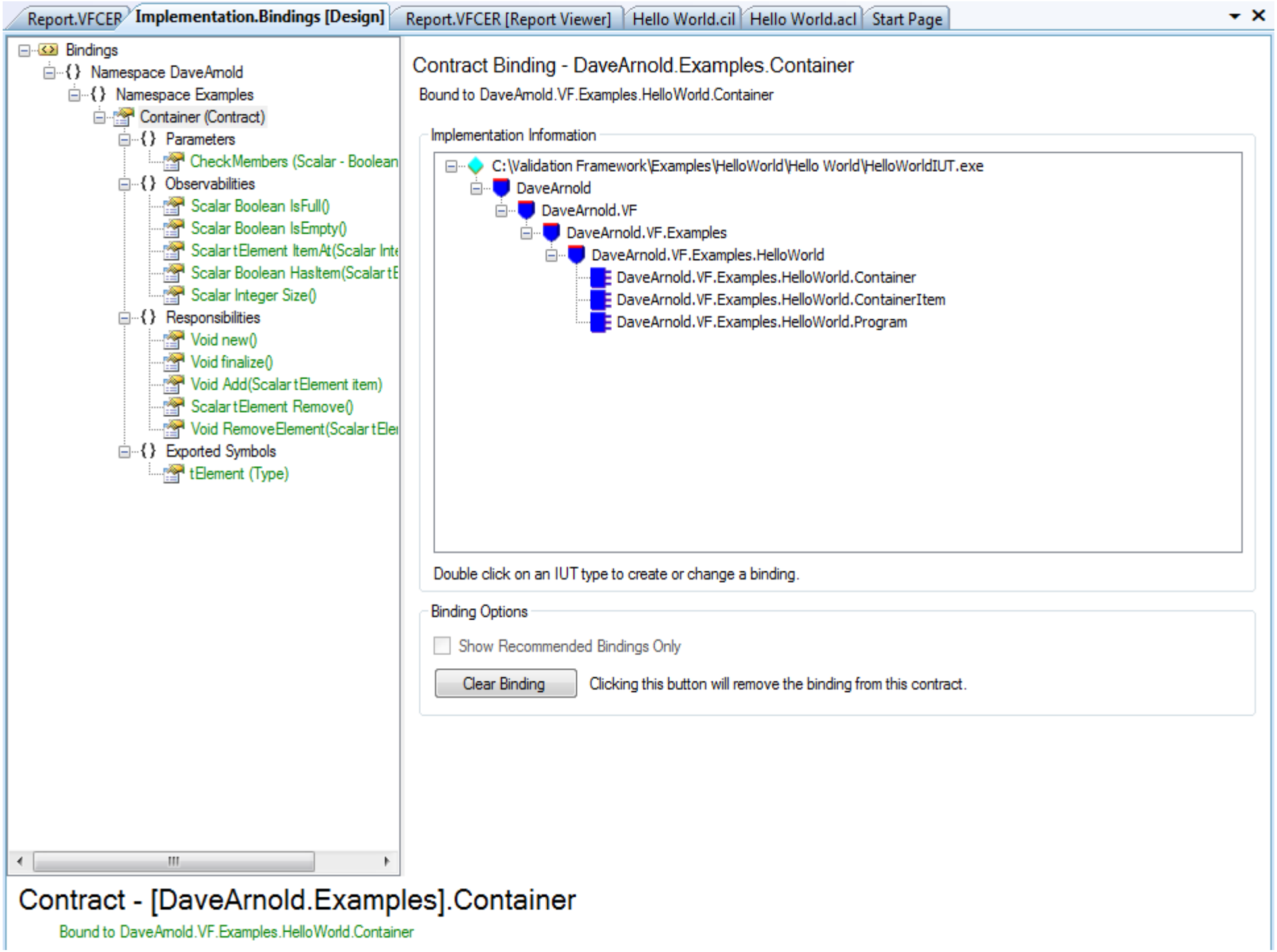
- Our TRM and supporting VF contribute in the areas of requirements engineering and validation by:

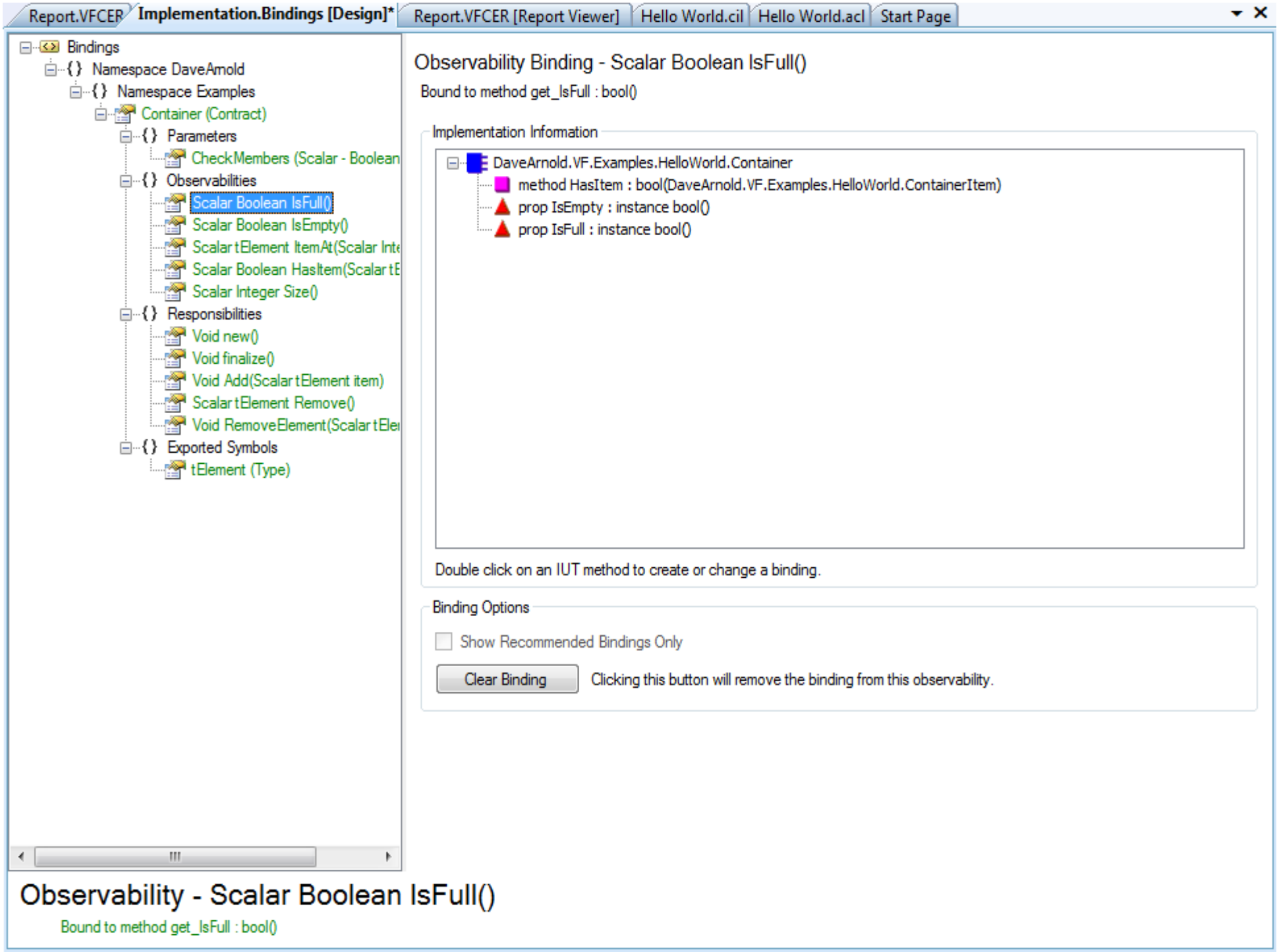
- Proposing a new set of requirements for a requirements model that supports operational validation. This set being the first, to the best of our knowledge, to include the following:
  - Capture of functional and non-functional requirements
  - Testability of the requirements model
  - Executabilty of checks generated from this testable model
  - Semantics rooted in the notions of responsibilities and scenarios
  - Abstraction of the testable model over several possible implementations
  - Openness to support specific static checks, dynamic checks, and metric evaluators
- Defining a TRM that satisfies these requirements (the ACL)
- Providing an open VF supporting the specification and execution of the TRM

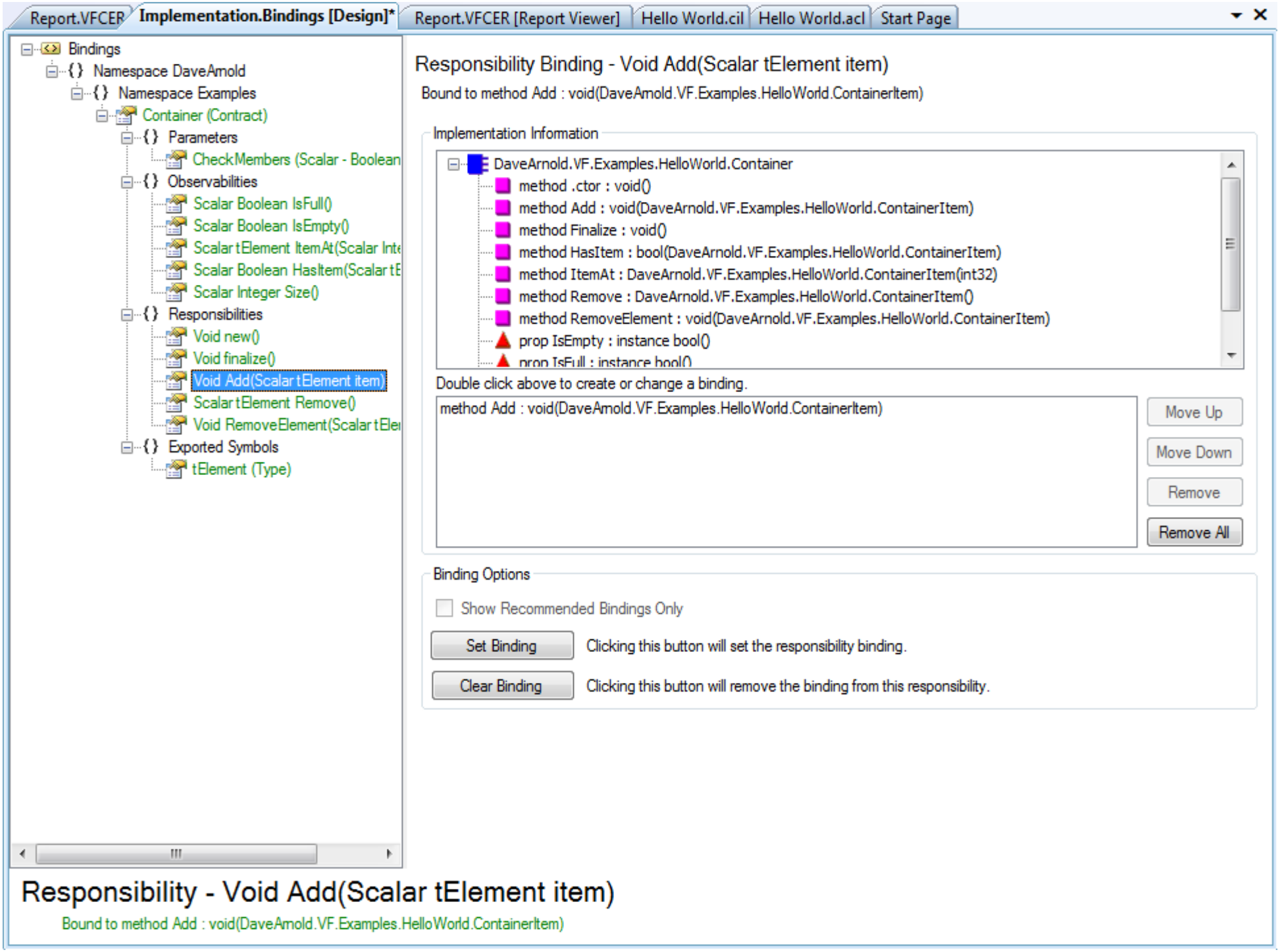
Future Work

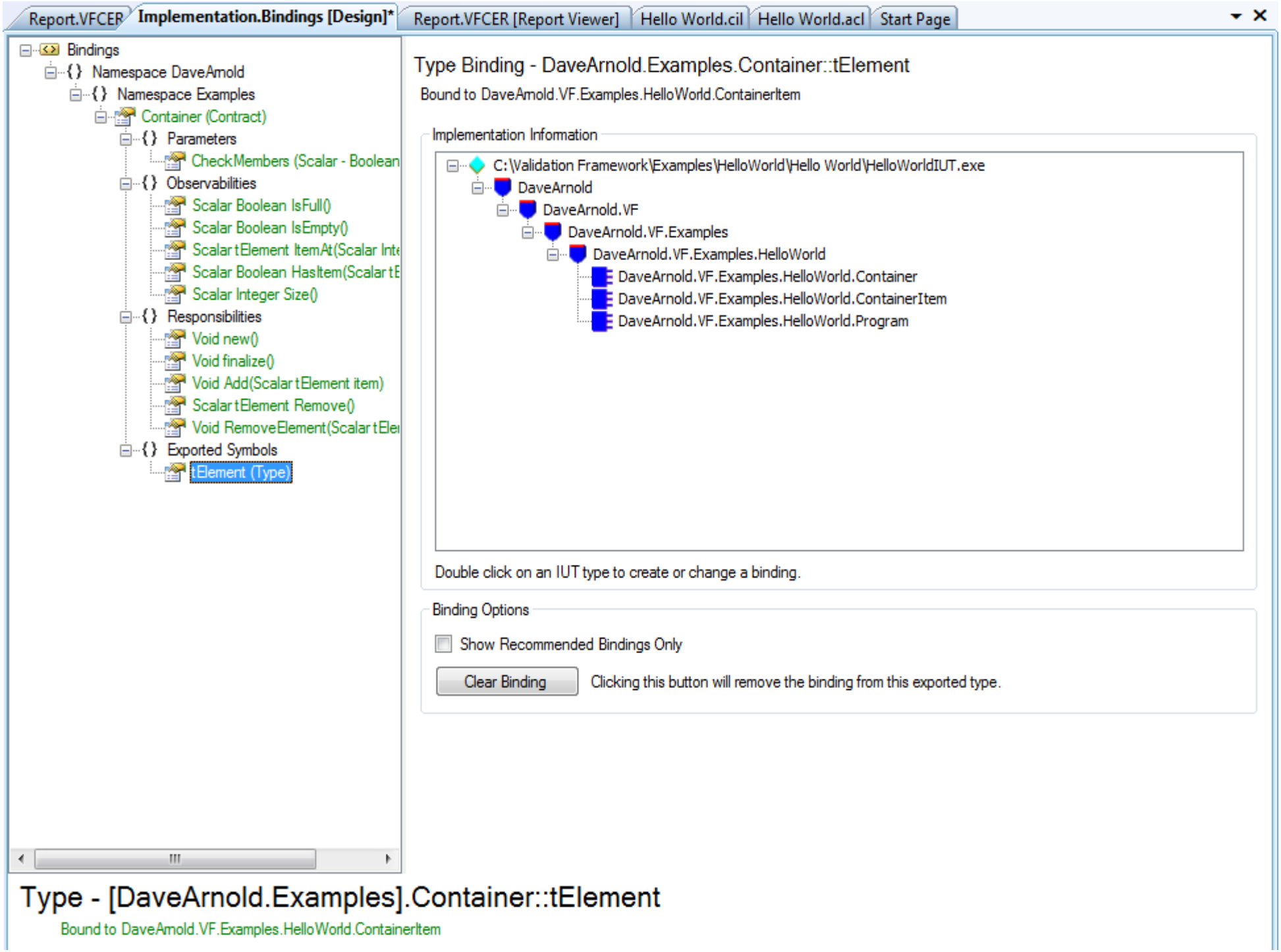
- Extension Through Openness
  - Additional high-level contract languages
    - Possibly domain specific
  - The creation of more AutoBind modules
  - The creation of more checks
    - Static checks
    - Dynamic checks
    - Metric evaluators


Validation Framework (Administrator)
File Edit View Project Build Tools Window Help
Build
CIL Runtime
115
Report.VFCER
Implementation.Bindings [Design]
Report.VFCER [Report Viewer]
Hello World.cil
Hello World.acl
Start Page
Bindings
Namespace DaveArnold
Namespace Examples
Container (Contract)
Parameters
CheckMembers (Scalar - Boolean
Observabilities
Scalar Boolean IsFull()
Scalar Boolean IsEmpty()
Scalar tElement ItemAt(Scalar Inte
Scalar Boolean HasItem(Scalar tE
Scalar Integer Size()
Responsibilities
Void new()
Void finalize()
Void Add(Scalar tElement item)
Scalar tElement Remove()
Void RemoveElement(Scalar tEle
Exported Symbols
tElement (Type)
In order to specifiy binding information select the ACL symbol from the left panel.
Depending on the selection made in the left panel, this panel will display valid IUT binding selections.
Binding Information
Bindings still required: 0
Bindings already specified: 13
Total bindings required: 13
Binding status: Bindings are complete. The contract project will now compile
Build Project
Clicking this button will build the contract project using the specified bindings.
Global Binding Options
Use "AutoBind" binding engine to automatically bind if possible.
The "AutoBind" binding engine will create automatic bindings if it is possible to infer the correct binding between the contract project and the IUT.
Selected "AutoBind" module - Default AutoBind Module [0.7.0.0] by Dave Arnold
Select "AutoBind" Module
Reset All Binding Information
Clicking this button will reset all binding information including parameter values.
All binding information will have to be re-specified
Bindings
Select an exported symbol to specify or view binding information.
Solution Explorer - Hello World
Solution 'Hello World' (1 project)
Hello World
Hello World.acl
HelloWorldIUT.exe
Implementation.Bindings
Report.VFCER
Loaded Plug-ins
Solution Explorer
Properties
Output
Show output from: General
CIL Project Support - Version 0.7.0.0 (Nov 2008) - Loaded.
Core Validation Framework Package - Version 0.7.0.0 (Nov 2008) - Loaded.
CIL Language Service and Compiler - Version 0.7.0.0 (Nov 2008) - Loaded.
Validation Framework Core Runtime 1.0 - Version 0.7.0.0 (Nov 2008) - Loaded.
ACL Language Service and Compiler - Version 0.7.0.0 (Nov 2008) - Loaded.
Error List
Output
Ready


Report.VFCER
Implementation.Bindings [Design]
Report.VFCER [Report Viewer]
Hello World.cil
Hello World.acl
Start Page
Bindings
Namespace DaveArnold
Namespace Examples
Container (Contract)
Parameters
CheckMembers (Scalar - Boolean
Observabilities
Scalar Boolean IsFull()
Scalar Boolean IsEmpty()
Scalar tElement ItemAt(Scalar Inte
Scalar Boolean HasItem(Scalar tE
Scalar Integer Size()
Responsibilities
Void new()
Void finalize()
Void Add(Scalar tElement item)
Scalar tElement Remove()
Void RemoveElement(Scalar tEle
Exported Symbols
tElement (Type)
Contract Binding - DaveArnold.Examples.Container
Bound to DaveArnold.VF.Examples.HelloWorld.Container
Implementation Information
C:\Validation Framework\Examples\HelloWorld\Hello World\HelloWorldIUT.exe
DaveArnold
DaveArnold.VF
DaveArnold.VF.Examples
DaveArnold.VF.Examples.HelloWorld
DaveArnold.VF.Examples.HelloWorld.Container
DaveArnold.VF.Examples.HelloWorld.ContainerItem
DaveArnold.VF.Examples.HelloWorld.Program
Double click on an IUT type to create or change a binding.
Binding Options
Show Recommended Bindings Only
Clear Binding
Clicking this button will remove the binding from this contract.
Contract - [DaveArnold.Examples].Container
Bound to DaveArnold.VF.Examples.HelloWorld.Container


Report.VFCER
Implementation.Bindings [Design]*
Report.VFCER [Report Viewer]
Hello World.cil
Hello World.acl
Start Page
Bindings
Namespace DaveArnold
Namespace Examples
Container (Contract)
Parameters
CheckMembers (Scalar - Boolean
Observabilities
Scalar Boolean IsFull()
Scalar Boolean IsEmpty()
Scalar tElement ItemAt(Scalar Inte
Scalar Boolean HasItem(Scalar tE
Scalar Integer Size()
Responsibilities
Void new()
Void finalize()
Void Add(Scalar tElement item)
Scalar tElement Remove()
Void RemoveElement(Scalar tEle
Exported Symbols
tElement (Type)
Observability Binding - Scalar Boolean IsFull()
Bound to method get_IsFull : bool()
Implementation Information
DaveArnold.VF.Examples.HelloWorld.Container
method HasItem : bool(DaveArnold.VF.Examples.HelloWorld.ContainerItem)
prop IsEmpty : instance bool()
prop IsFull : instance bool()
Double click on an IUT method to create or change a binding.
Binding Options
Show Recommended Bindings Only
Clear Binding
Clicking this button will remove the binding from this observability.
Observability - Scalar Boolean IsFull()
Bound to method get_IsFull : bool()

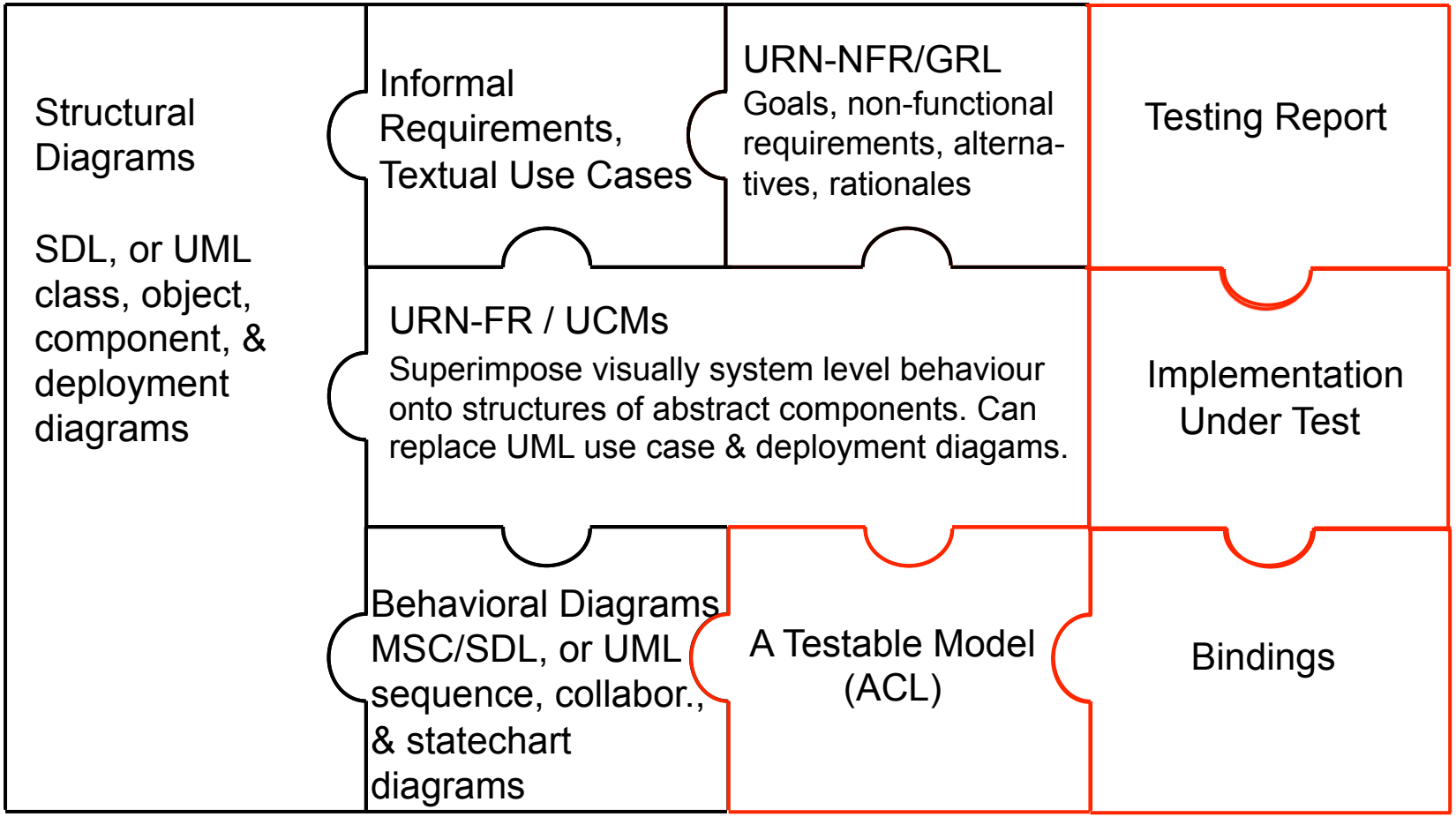

Report.VFCER
Implementation.Bindings [Design]*
Report.VFCER [Report Viewer]
Hello World.cil
Hello World.acl
Start Page
Bindings
Namespace DaveArnold
Namespace Examples
Container (Contract)
Parameters
CheckMembers (Scalar - Boolean
Observabilities
Scalar Boolean IsFull()
Scalar Boolean IsEmpty()
Scalar tElement ItemAt(Scalar Inte
Scalar Boolean HasItem(Scalar tE
Scalar Integer Size()
Responsibilities
Void new()
Void finalize()
Void Add(Scalar tElement item)
Scalar tElement Remove()
Void RemoveElement(Scalar tEle
Exported Symbols
tElement (Type)
Responsibility Binding - Void Add(Scalar tElement item)
Bound to method Add : void(DaveArnold.VF.Examples.HelloWorld.ContainerItem)
Implementation Information
DaveArnold.VF.Examples.HelloWorld.Container
method .ctor : void()
method Add : void(DaveArnold.VF.Examples.HelloWorld.ContainerItem)
method Finalize : void()
method HasItem : bool(DaveArnold.VF.Examples.HelloWorld.ContainerItem)
method ItemAt : DaveArnold.VF.Examples.HelloWorld.ContainerItem(int32)
method Remove : DaveArnold.VF.Examples.HelloWorld.ContainerItem()
method RemoveElement : void(DaveArnold.VF.Examples.HelloWorld.ContainerItem)
prop IsEmpty : instance bool()
prop IsFull : instance bool()
Double click above to create or change a binding.
method Add : void(DaveArnold.VF.Examples.HelloWorld.ContainerItem)
Move Up
Move Down
Remove
Remove All
Binding Options
Show Recommended Bindings Only
Set Binding
Clicking this button will set the responsibility binding.
Clear Binding
Clicking this button will remove the binding from this responsibility.
Responsibility - Void Add(Scalar tElement item)
Bound to method Add : void(DaveArnold.VF.Examples.HelloWorld.ContainerItem)

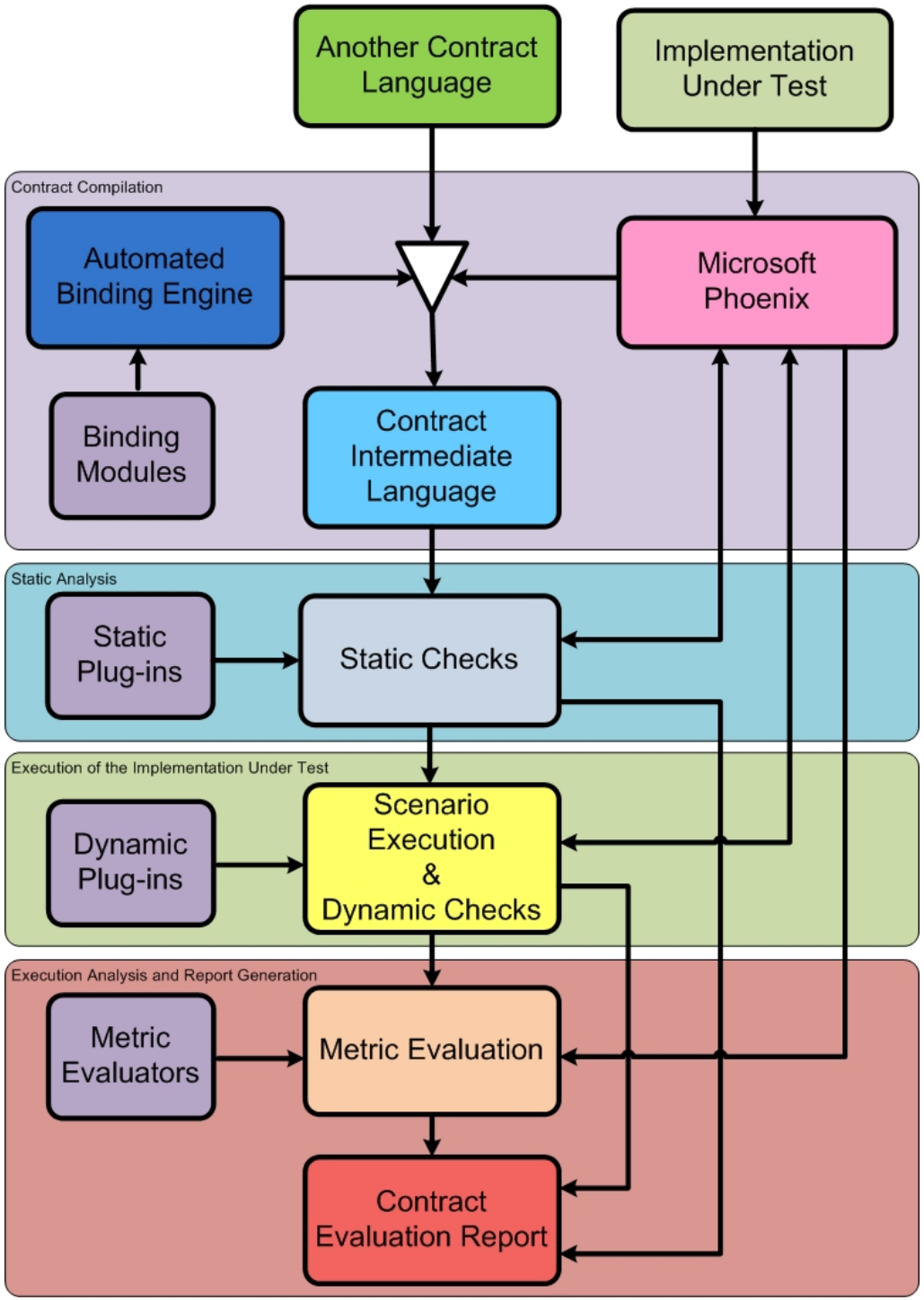

Report.VFCER
Implementation.Bindings [Design]*
Report.VFCER [Report Viewer]
Hello World.cil
Hello World.acl
Start Page
Bindings
Namespace DaveArnold
Namespace Examples
Container (Contract)
Parameters
CheckMembers (Scalar - Boolean
Observabilities
Scalar Boolean IsFull()
Scalar Boolean IsEmpty()
Scalar tElement ItemAt(Scalar Inte
Scalar Boolean HasItem(Scalar tE
Scalar Integer Size()
Responsibilities
Void new()
Void finalize()
Void Add(Scalar tElement item)
Scalar tElement Remove()
Void RemoveElement(Scalar tEle
Exported Symbols
tElement (Type)
Type Binding - DaveArnold.Examples.Container::tElement
Bound to DaveArnold.VF.Examples.HelloWorld.ContainerItem
Implementation Information
C:\Validation Framework\Examples\HelloWorld\Hello World\HelloWorldIUT.exe
DaveArnold
DaveArnold.VF
DaveArnold.VF.Examples
DaveArnold.VF.Examples.HelloWorld
DaveArnold.VF.Examples.HelloWorld.Container
DaveArnold.VF.Examples.HelloWorld.ContainerItem
DaveArnold.VF.Examples.HelloWorld.Program
Double click on an IUT type to create or change a binding.
Binding Options
Show Recommended Bindings Only
Clear Binding
Clicking this button will remove the binding from this exported type.
Type - [DaveArnold.Examples].Container::tElement
Bound to DaveArnold.VF.Examples.HelloWorld.ContainerItem

CIL Generation

- Once the binding process is complete, the ACL and binding tables are used to generate a Contract Intermediate Language (CIL) representation.
  - Low level stack-based language
  - Designed so that other high level contract languages can be used with the runtime
    - Possibly graphical representations

Scope of our Work




Another Contract Language
Implementation Under Test
Contract Compilation
Automated Binding Engine
Microsoft Phoenix
Binding Modules
Contract Intermediate Language
Static Analysis
Static Plug-ins
Static Checks
Execution of the Implementation Under Test
Dynamic Plug-ins
Scenario Execution & Dynamic Checks
Execution Analysis and Report Generation
Metric Evaluators
Metric Evaluation
Contract Evaluation Report

On Capturing Requirements

We distinguish 3 ‘schools’:

- Formal
  - Require hard-to-find expertise
  - Unified? Executable? Traceable to code?
- Code-based
  - Modeling is minimized => no testable model
  - Agile methods (e.g., TDD) advocate intensive collaboration
- Model-Based
  - Testable? Unified? Executable?
  - Full code generation DOES require implementation-aware designers!