Spec Explorer

A Model-Based Testing tool from Microsoft

- Manuel Naujoks
- IM2
- http://mnsdc.de
- http://twitter.com/halllo

Inhalt

Modellbasiertes Testen (MBT)

MBT Probleme

Von Abstract State Machines bis Spec Explorer

Spec Explorer 2010

Einsatz bei Microsoft

Fazit

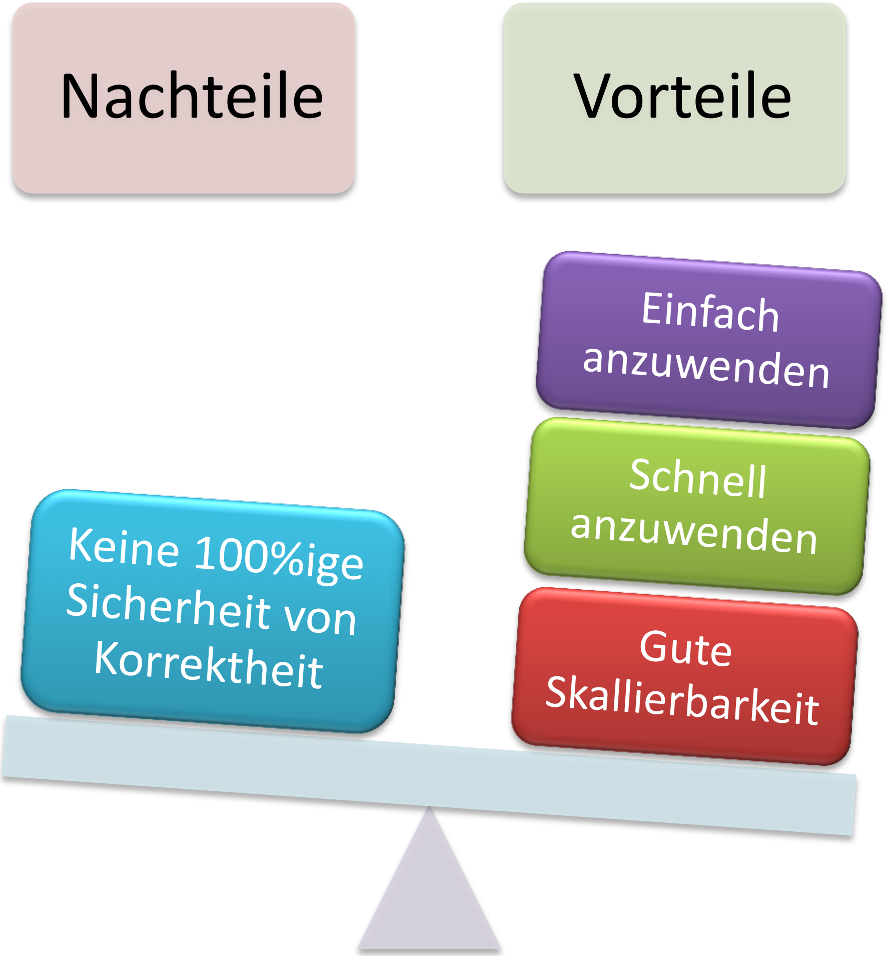
Spec Explorer

Modellbasiertes Testen (MBT)

Modellbasiertes Testen (MBT)

- Formal
  - Maschinenlesbare Spezifikation
- Leichtgewichtig
  - Kein mathematischer Beweis erforderlich
  - Hohe Wahrscheinlichkeit von Korrektheit

Modellbasiertes Testen (MBT)



Nachteile

Vorteile

Einfach anzuwenden

Schnell anzuwenden

Gute Skallierbarkeit

Keine 100%ige Sicherheit von Korrektheit

Spec Explorer

MBT Probleme

MBT Probleme

- Multi-paradigmatic Model-Based Testing (Springer-Verlag, 2006)
  - **Wolfgang Grieskamp**
  - http://www.springerlink.com/content/087hx3160357088u

MBT Probleme

- State Explosion Problem
- Proprietäre Spezifikationssprache erforderlich (anstelle einer Mainstream Sprache)
- Keine Unterstützung für szenarienorientiertes Modellieren (nur zustandsorientiert)
- Keine IDE Unterstützung
- Keine Integration mit bestehenden Test Frameworks & Runnern

Spec Explorer

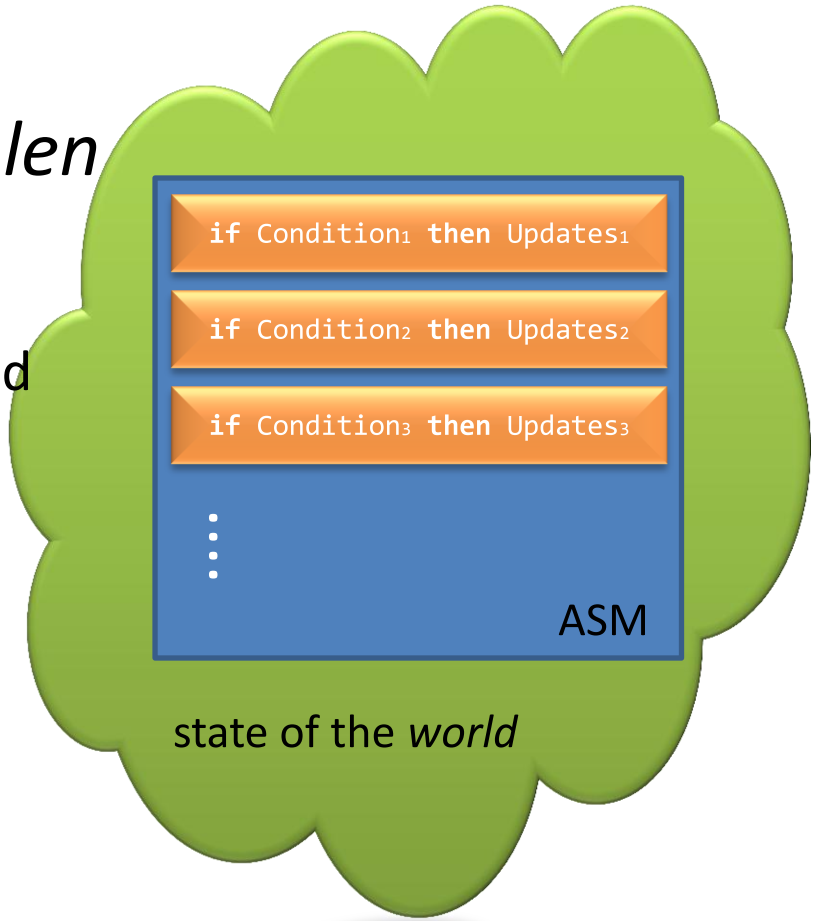
Von Abstract State Machines bis Spec Explorer

Von ASMs bis Spec Explorer

- Yuri Gurevich: *Modellierung aller Algorithmem möglich?*
- Ja, wenn Zustände des Algorithmus alle relevanten Informationen enthalten
- Kleiner Befehlssatz in allen Fällen ausreichend: Abstract State Machine (ASM)
- **Yuri Gurevich**, 1999, *Sequential Abstract State Machines Capture Sequential Algorithms*
  - ftp://www.eecs.umich.edu/groups/gasm/seqthesis.pdf

Von ASMs bis Spec Explorer

- Aktueller Zustand der Maschine ist die Menge der aktuellen Werte aller *Variablen*
  - Condition
    - Prädikat, prüft anhand der *Variablen* auf konkreten Zustand
  - Updates
    - Menge an Funktionen, die die *Variablen* verändern
- Erklärung nach **Egon Börger**
  - zusammen mit Robert Stärk: AsmBook (Springer-Verlag, 2003)



if Condition$_1$ **then** Updates$_1$

if Condition$_2$ **then** Updates$_2$

if Condition$_3$ **then** Updates$_3$

⋮

ASM

state of the *world*

Von ASMs bis Spec Explorer

- Specification language AsmL (Abstract State Machine Language)
  - **Yuri Gurevich**, **Wolfgang Grieskamp**, Margus Veanes, Wolfram Schulte, Lev Nachmanson, Colin Campbell, **Nikolai Tillmann**
  - http://asml.codeplex.com/
- state space exploration
  - **Wolfgang Grieskamp** & **Nikolai Tillmann**
  - first Model-Based testing tool, AsmL-T

Von ASMs bis Spec Explorer

- AsmL-T wurde genutzt Indigo (WCF) zu testen
- Testen mit AsmL-T wurde beliebter als AsmL
- Light weight AsmL aka **Spec#**
  - code contracts
  - AsmL in C#-Syntax
- Neuer Name für AsmL-T: **Spec Explorer 2004**
- 2005 Teamtrennung: PEX und MBT

Von ASMs bis Spec Explorer

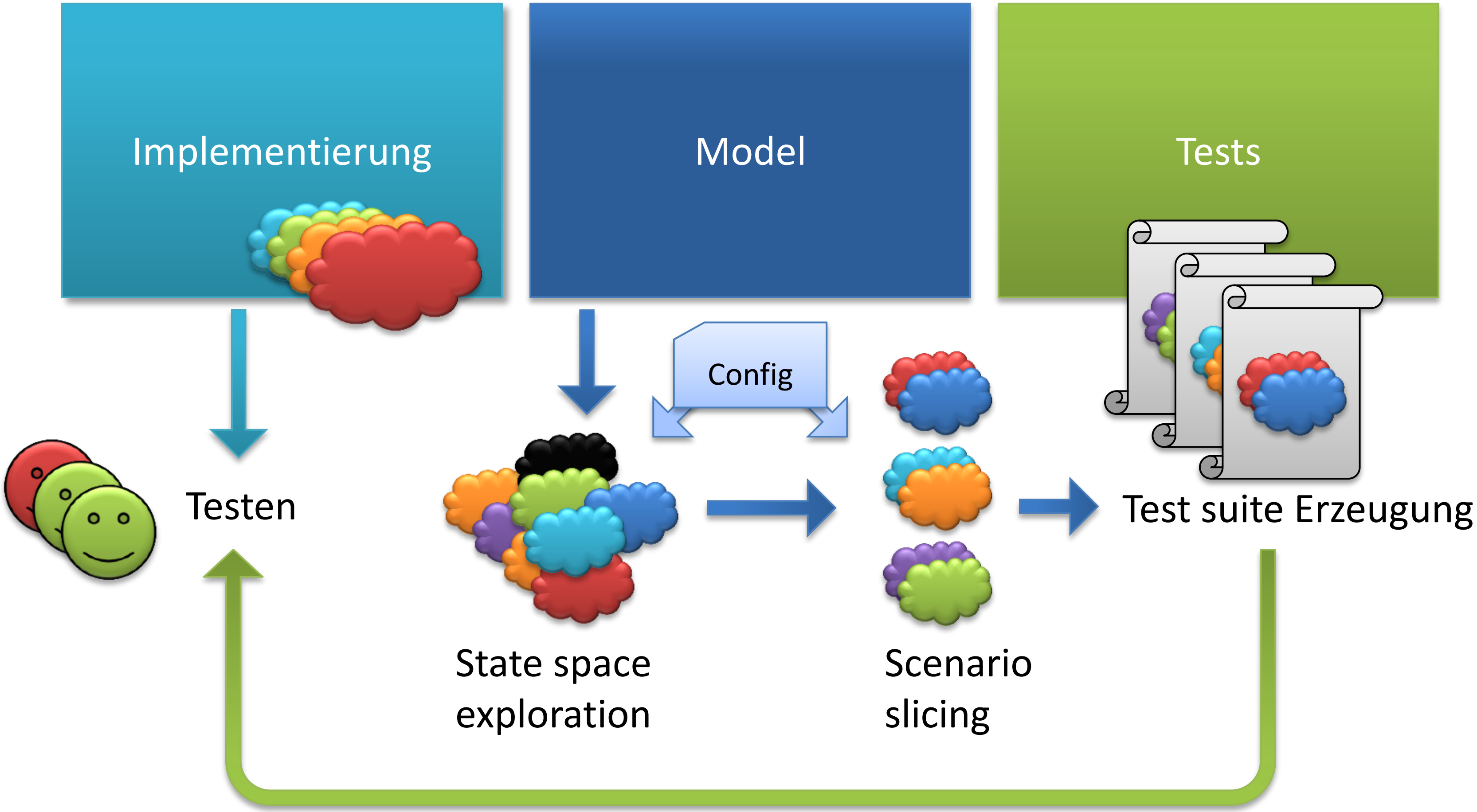
- Model-based Software Testing and Analysis with C# (Cambridge University Press, 2008)
  - Jonathan Jacky, Margus Veanes, Colin Campbell, Wolfram Schulte
  - http://staff.washington.edu/jon/modeling-book/
  - Open-source implementation called **NModel** http://nmodel.codeplex.com/

Spec Explorer 2010

Spec Explorer 2010

- Die Lösung
  - Erster Prototype 2006
  - Model exploration auf der CIL (C# as input notation)
  - Scripting language CORD
  - Visual Studio integration
  - Automatische Test suite Erzeugung für Microsoft.VisualStudio.TestTools.UnitTesting

Spec Explorer 2010



Implementierung

Model

Tests

Config

Testen

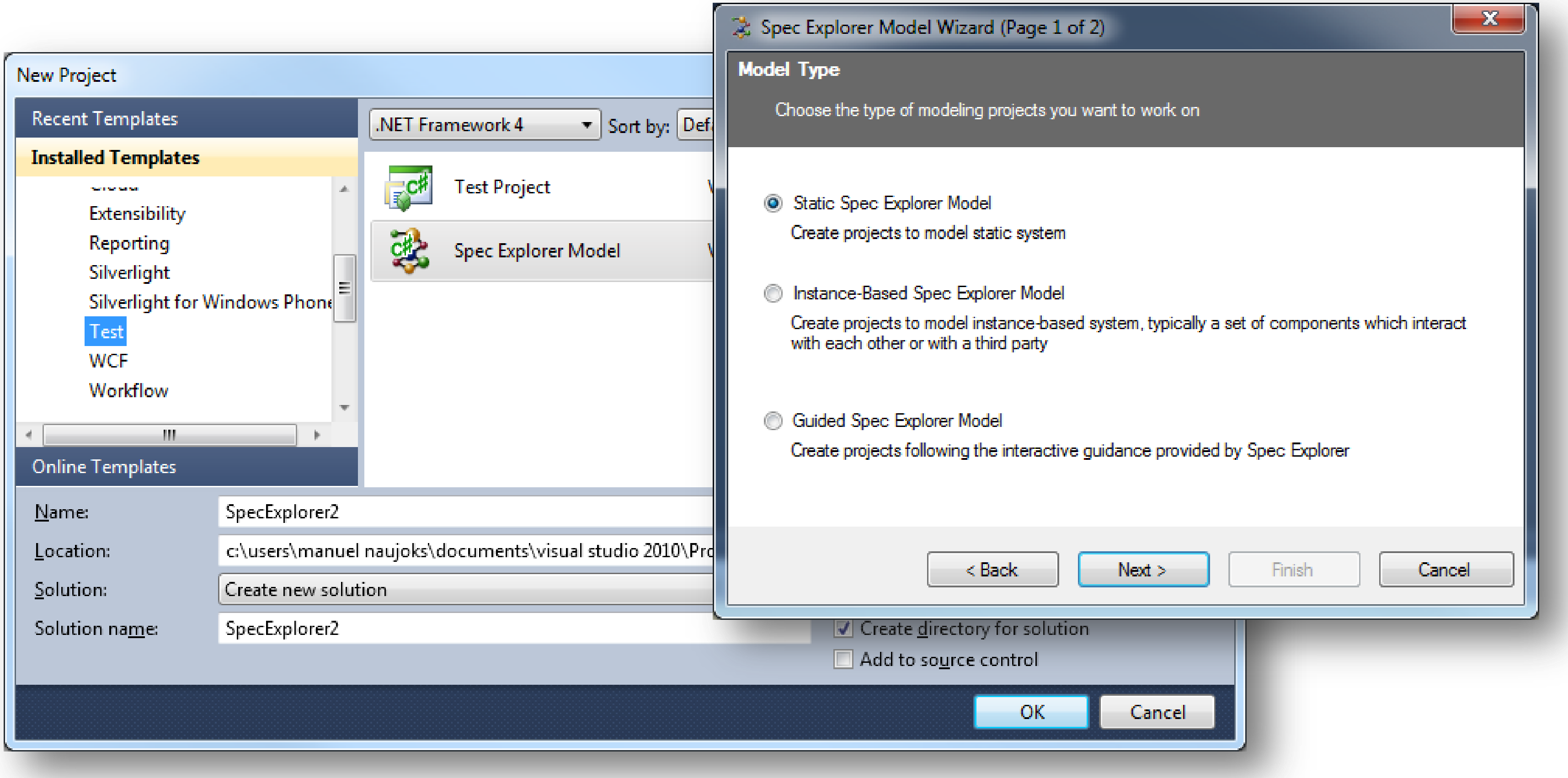
State space exploration

Scenario slicing

Test suite Erzeugung

Beispiel

- File / New Project



New Project

Recent Templates

Installed Templates

Extensibility

Reporting

Silverlight

Silverlight for Windows Phone

Test

WCF

Workflow

Online Templates

.NET Framework 4 Sort by: Def

Test Project

Spec Explorer Model

Name: SpecExplorer2

Location: c:\users\manuel naujoks\documents\visual studio 2010\Pro

Solution: Create new solution

Solution name: SpecExplorer2

☑ Create directory for solution

☐ Add to source control

OK Cancel

Spec Explorer Model Wizard (Page 1 of 2)

Model Type

Choose the type of modeling projects you want to work on

● Static Spec Explorer Model

Create projects to model static system

○ Instance-Based Spec Explorer Model

Create projects to model instance-based system, typically a set of components which interact with each other or with a third party

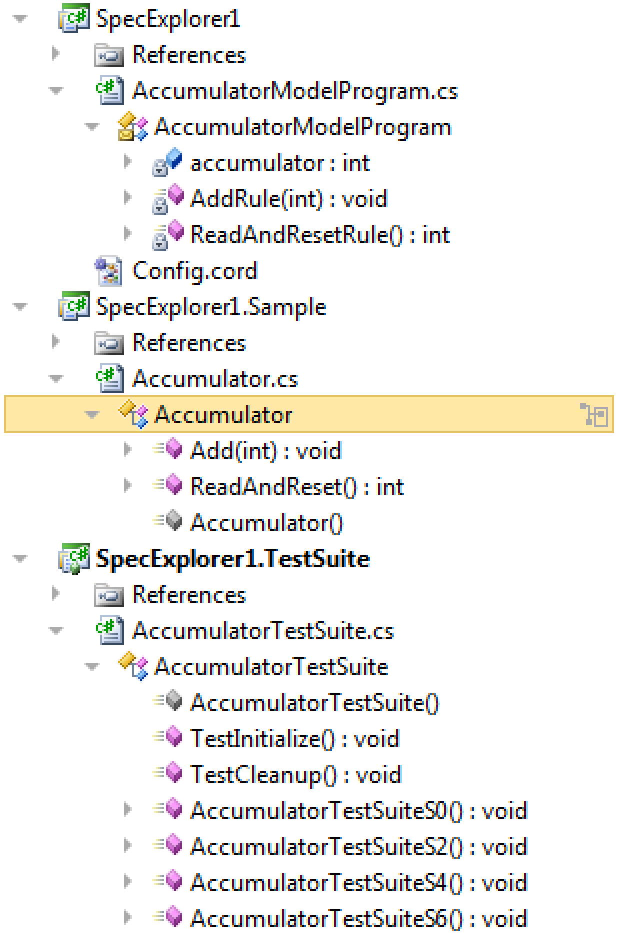
○ Guided Spec Explorer Model

Create projects following the interactive guidance provided by Spec Explorer

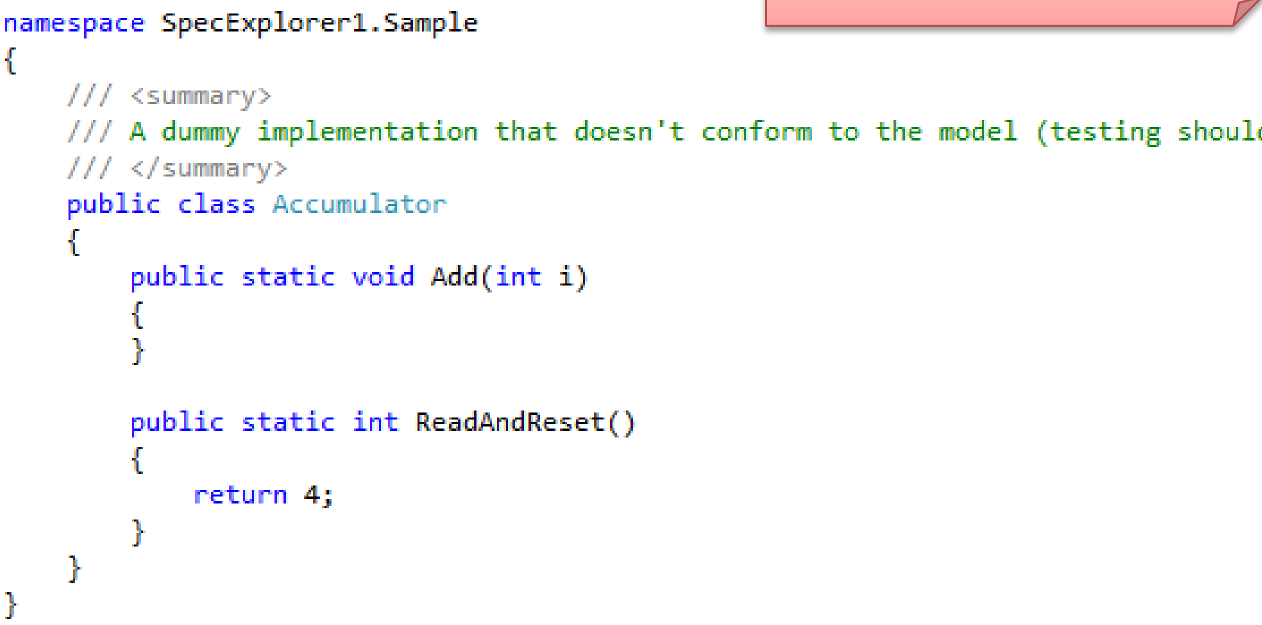
< Back Next > Finish Cancel

Implementierung

Fehlerhaft

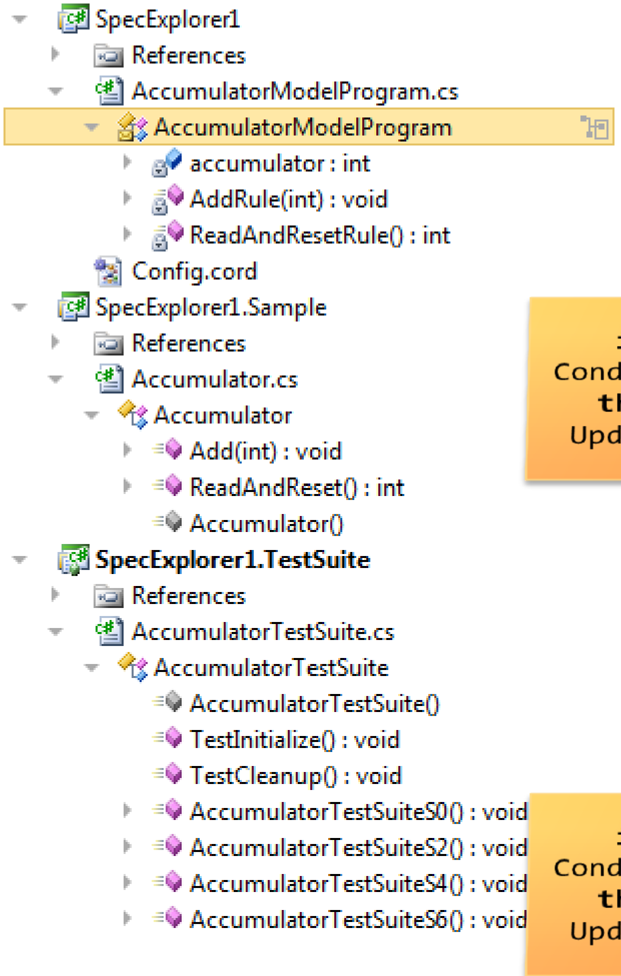


- SpecExplorer1
  - References
  - AccumulatorModelProgram.cs
    - AccumulatorModelProgram
      - accumulator : int
      - AddRule(int) : void
      - ReadAndResetRule() : int
  - Config.cord
- SpecExplorer1.Sample
  - References
  - Accumulator.cs
    - Accumulator
      - Add(int) : void
      - ReadAndReset() : int
      - Accumulator()
- **SpecExplorer1.TestSuite**
  - References
  - AccumulatorTestSuite.cs
    - AccumulatorTestSuite
      - AccumulatorTestSuite()
      - TestInitialize() : void
      - TestCleanup() : void
      - AccumulatorTestSuiteS0() : void
      - AccumulatorTestSuiteS2() : void
      - AccumulatorTestSuiteS4() : void
      - AccumulatorTestSuiteS6() : void

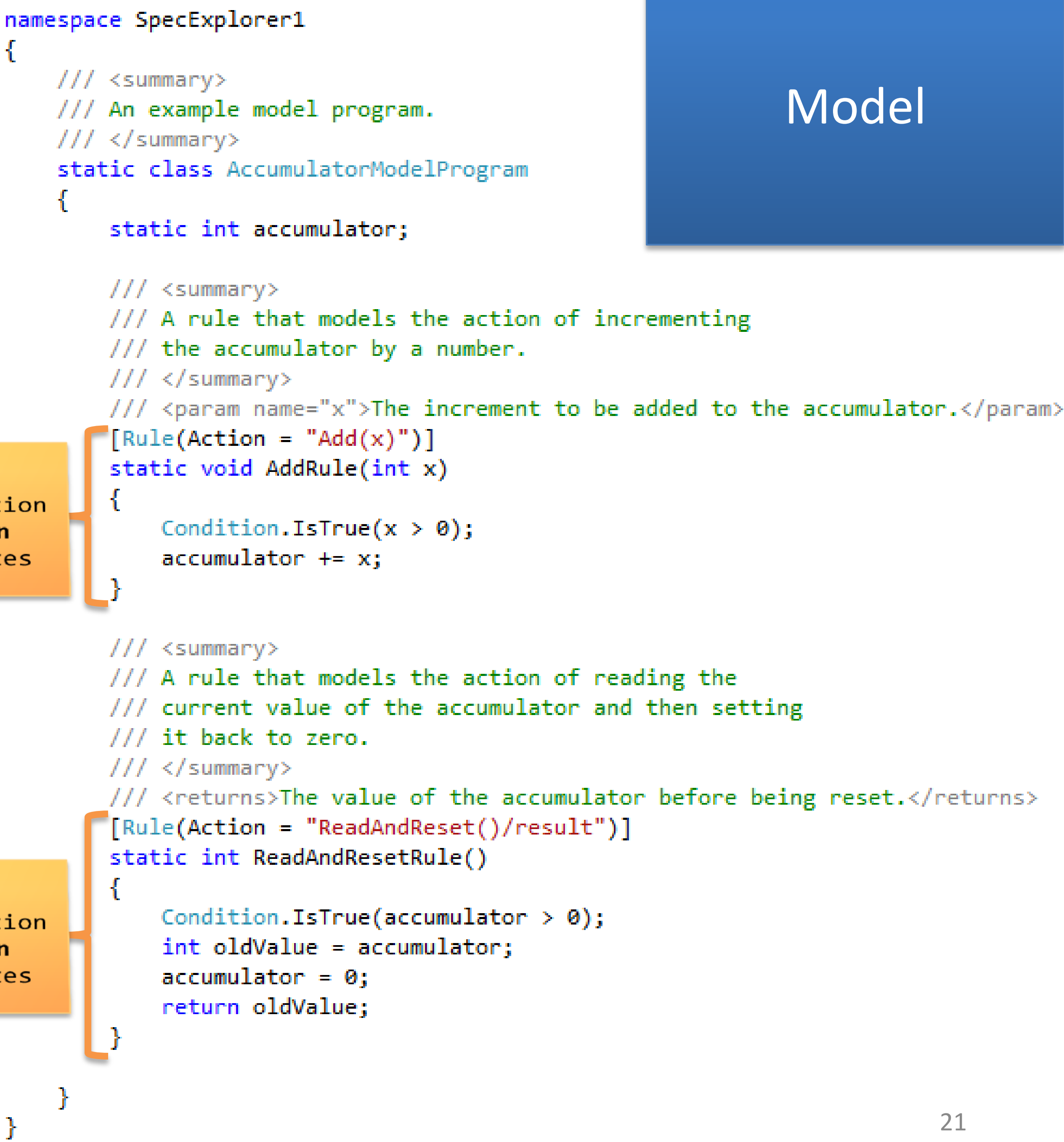


```
namespace SpecExplorer1.Sample
{
    /// <summary>
    /// A dummy implementation that doesn't conform to the model (testing shoul
    /// </summary>
    public class Accumulator
    {
        public static void Add(int i)
        {
        }

        public static int ReadAndReset()
        {
            return 4;
        }
    }
}
```

Model



```
SpecExplorer1
    References
    AccumulatorModelProgram.cs
        AccumulatorModelProgram
            accumulator : int
            AddRule(int) : void
            ReadAndResetRule() : int
    Config.cord
SpecExplorer1.Sample
    References
    Accumulator.cs
        Accumulator
            Add(int) : void
            ReadAndReset() : int
            Accumulator()
SpecExplorer1.TestSuite
    References
    AccumulatorTestSuite.cs
        AccumulatorTestSuite
            AccumulatorTestSuite()
            TestInitialize() : void
            TestCleanup() : void
            AccumulatorTestSuiteS0() : void
            AccumulatorTestSuiteS2() : void
            AccumulatorTestSuiteS4() : void
            AccumulatorTestSuiteS6() : void
```



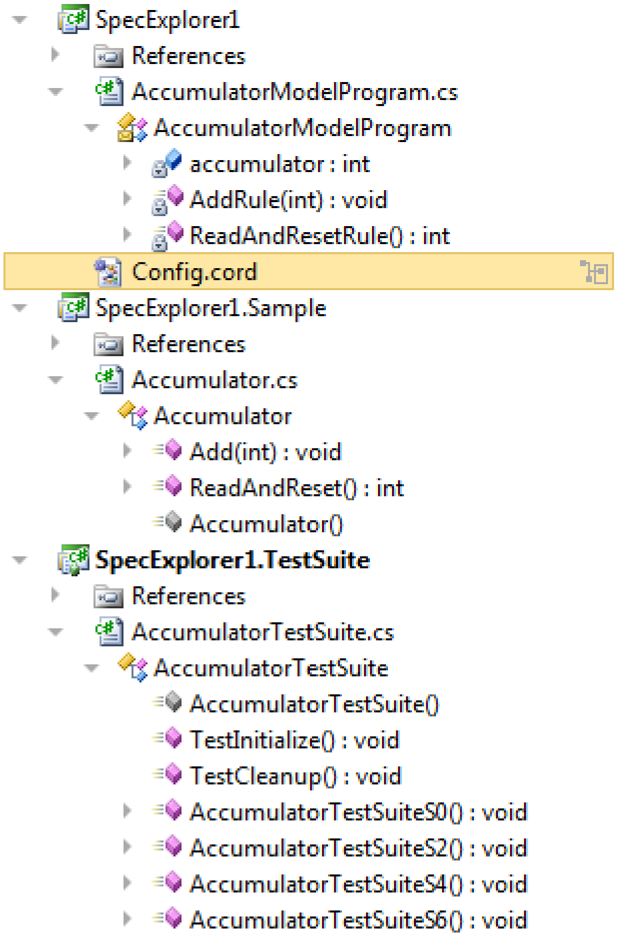
```
namespace SpecExplorer1
{
    /// <summary>
    /// An example model program.
    /// </summary>
    static class AccumulatorModelProgram
    {
        static int accumulator;

        /// <summary>
        /// A rule that models the action of incrementing
        /// the accumulator by a number.
        /// </summary>
        /// <param name="x">The increment to be added to the accumulator.</param>
        [Rule(Action = "Add(x)")]
        static void AddRule(int x)
        {
            Condition.IsTrue(x > 0);
            accumulator += x;
        }

        /// <summary>
        /// A rule that models the action of reading the
        /// current value of the accumulator and then setting
        /// it back to zero.
        /// </summary>
        /// <returns>The value of the accumulator before being reset.</returns>
        [Rule(Action = "ReadAndReset()/result")]
        static int ReadAndResetRule()
        {
            Condition.IsTrue(accumulator > 0);
            int oldValue = accumulator;
            accumulator = 0;
            return oldValue;
        }

    }
}
```

if
Condition
then
Updates

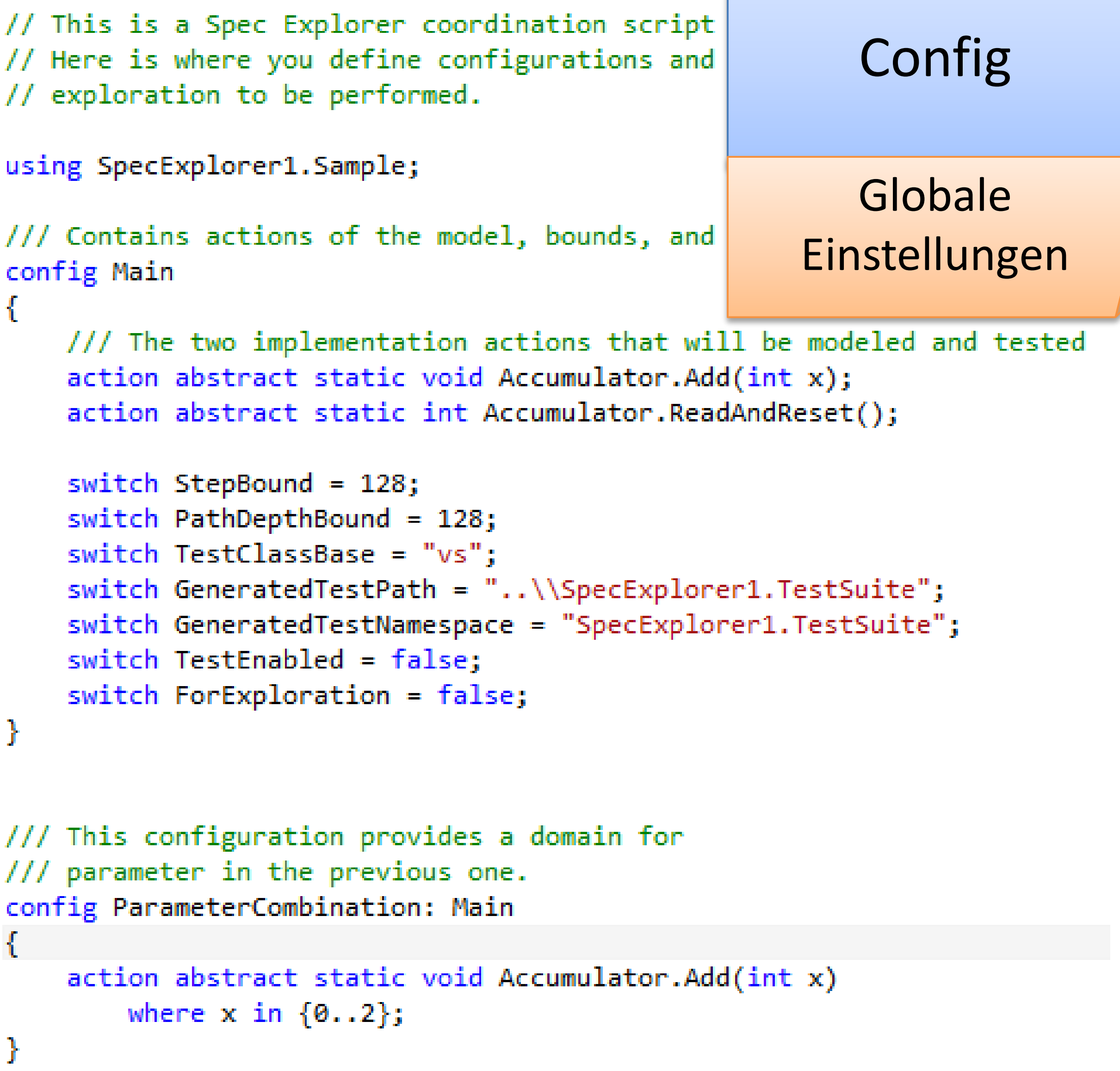
if
Condition
then
Updates

Config

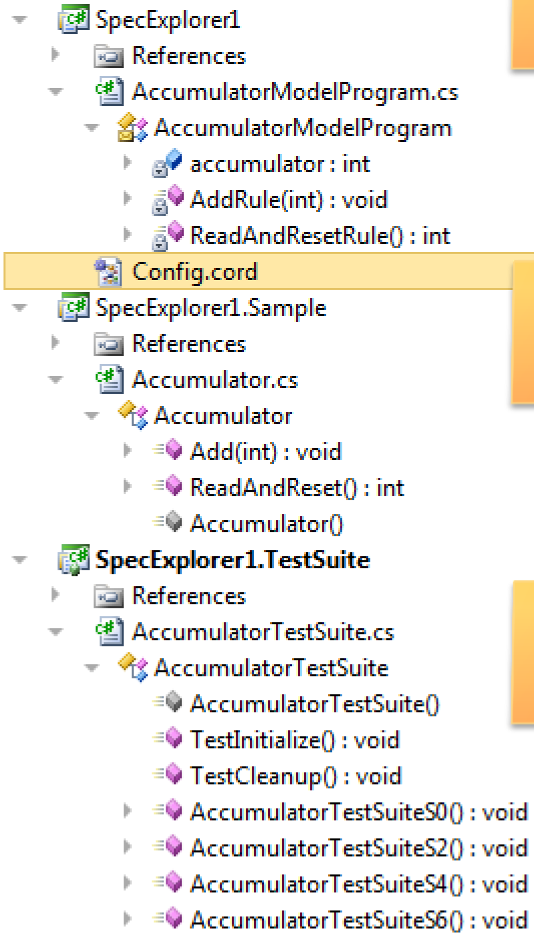
Globale Einstellungen



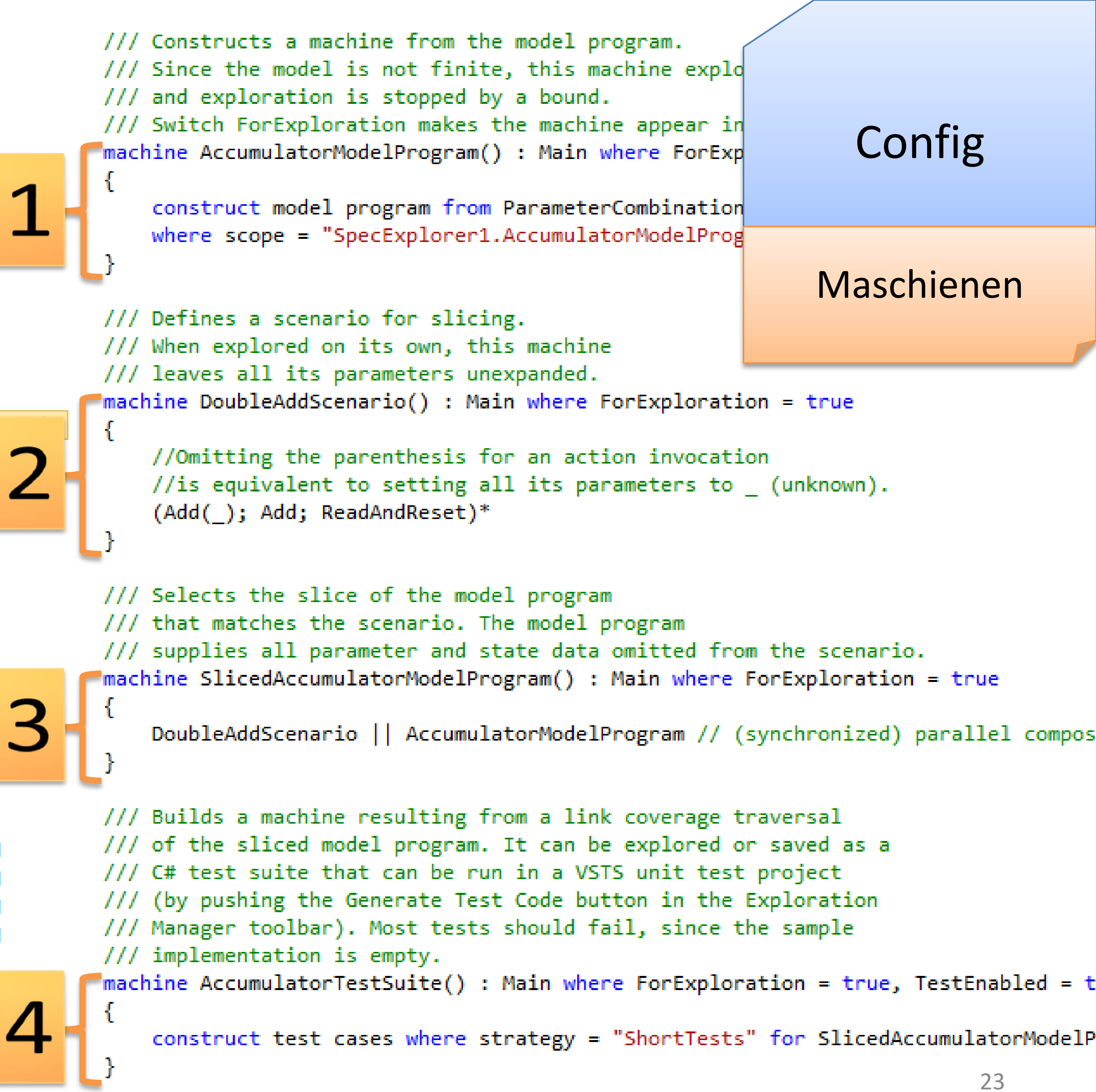
```
SpecExplorer1
    References
    AccumulatorModelProgram.cs
        AccumulatorModelProgram
            accumulator : int
            AddRule(int) : void
            ReadAndResetRule() : int
    Config.cord
SpecExplorer1.Sample
    References
    Accumulator.cs
        Accumulator
            Add(int) : void
            ReadAndReset() : int
            Accumulator()
SpecExplorer1.TestSuite
    References
    AccumulatorTestSuite.cs
        AccumulatorTestSuite
            AccumulatorTestSuite()
            TestInitialize() : void
            TestCleanup() : void
            AccumulatorTestSuiteS0() : void
            AccumulatorTestSuiteS2() : void
            AccumulatorTestSuiteS4() : void
            AccumulatorTestSuiteS6() : void
```



```
// This is a Spec Explorer coordination script
// Here is where you define configurations and
// exploration to be performed.

using SpecExplorer1.Sample;

/// Contains actions of the model, bounds, and
config Main
{
    /// The two implementation actions that will be modeled and tested
    action abstract static void Accumulator.Add(int x);
    action abstract static int Accumulator.ReadAndReset();

    switch StepBound = 128;
    switch PathDepthBound = 128;
    switch TestClassBase = "vs";
    switch GeneratedTestPath = "..\\SpecExplorer1.TestSuite";
    switch GeneratedTestNamespace = "SpecExplorer1.TestSuite";
    switch TestEnabled = false;
    switch ForExploration = false;
}


/// This configuration provides a domain for
/// parameter in the previous one.
config ParameterCombination: Main
{
    action abstract static void Accumulator.Add(int x)
        where x in {0..2};
}
```



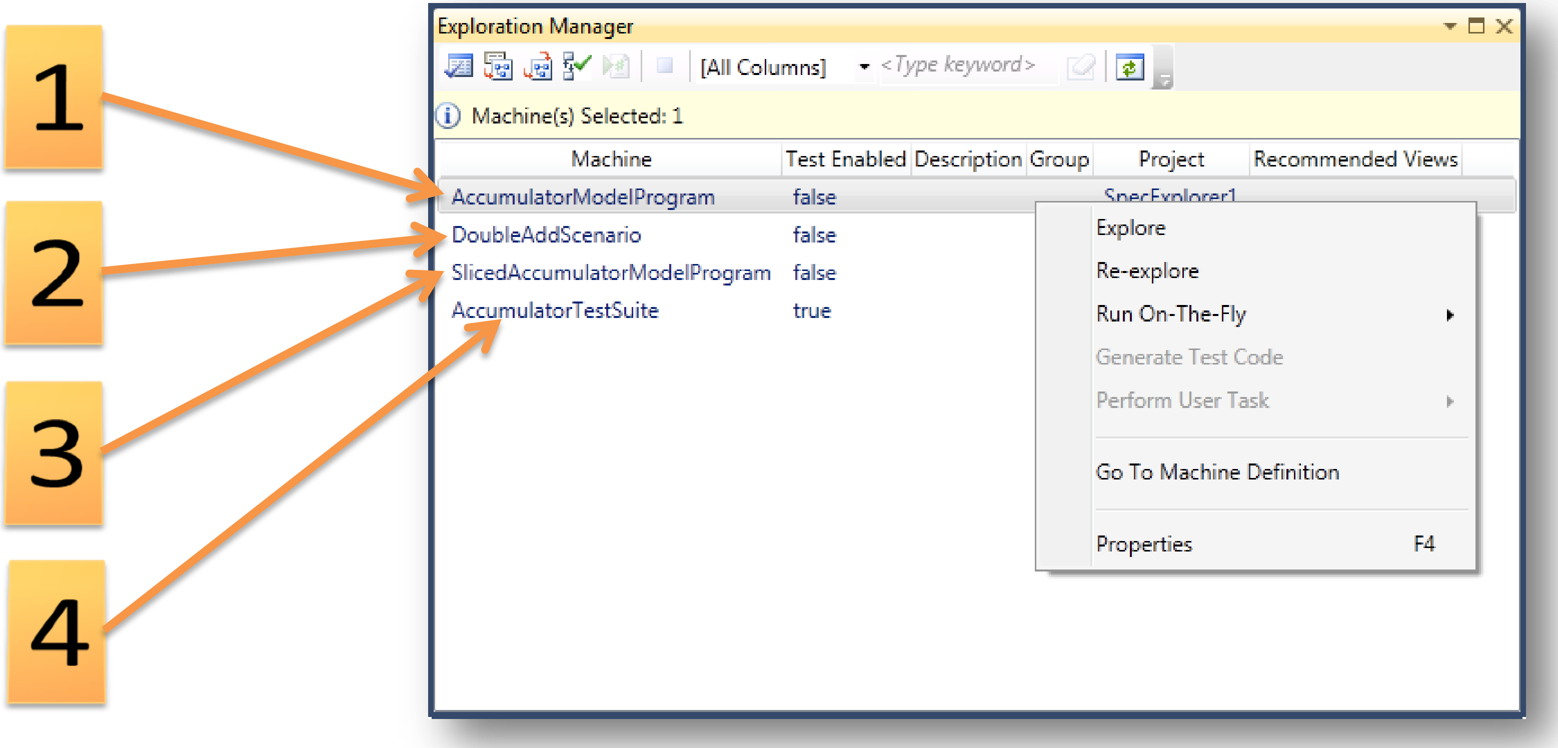
```
SpecExplorer1
    References
    AccumulatorModelProgram.cs
        AccumulatorModelProgram
            accumulator : int
            AddRule(int) : void
            ReadAndResetRule() : int
    Config.cord
SpecExplorer1.Sample
    References
    Accumulator.cs
        Accumulator
            Add(int) : void
            ReadAndReset() : int
            Accumulator()
SpecExplorer1.TestSuite
    References
    AccumulatorTestSuite.cs
        AccumulatorTestSuite
            AccumulatorTestSuite()
            TestInitialize() : void
            TestCleanup() : void
            AccumulatorTestSuiteS0() : void
            AccumulatorTestSuiteS2() : void
            AccumulatorTestSuiteS4() : void
            AccumulatorTestSuiteS6() : void
```



```
/// Constructs a machine from the model program.
/// Since the model is not finite, this machine explo
/// and exploration is stopped by a bound.
/// Switch ForExploration makes the machine appear in
machine AccumulatorModelProgram() : Main where ForExp
{
    construct model program from ParameterCombination
    where scope = "SpecExplorer1.AccumulatorModelProg
}

/// Defines a scenario for slicing.
/// When explored on its own, this machine
/// leaves all its parameters unexpanded.
machine DoubleAddScenario() : Main where ForExploration = true
{
    //Omitting the parenthesis for an action invocation
    //is equivalent to setting all its parameters to _ (unknown).
    (Add(_); Add; ReadAndReset)*
}

/// Selects the slice of the model program
/// that matches the scenario. The model program
/// supplies all parameter and state data omitted from the scenario.
machine SlicedAccumulatorModelProgram() : Main where ForExploration = true
{
    DoubleAddScenario || AccumulatorModelProgram // (synchronized) parallel compos
}

/// Builds a machine resulting from a link coverage traversal
/// of the sliced model program. It can be explored or saved as a
/// C# test suite that can be run in a VSTS unit test project
/// (by pushing the Generate Test Code button in the Exploration
/// Manager toolbar). Most tests should fail, since the sample
/// implementation is empty.
machine AccumulatorTestSuite() : Main where ForExploration = true, TestEnabled = t
{
    construct test cases where strategy = "ShortTests" for SlicedAccumulatorModelP
}
```

Exploration Manager

- Ausflistung aller Maschienen aus Config.cord



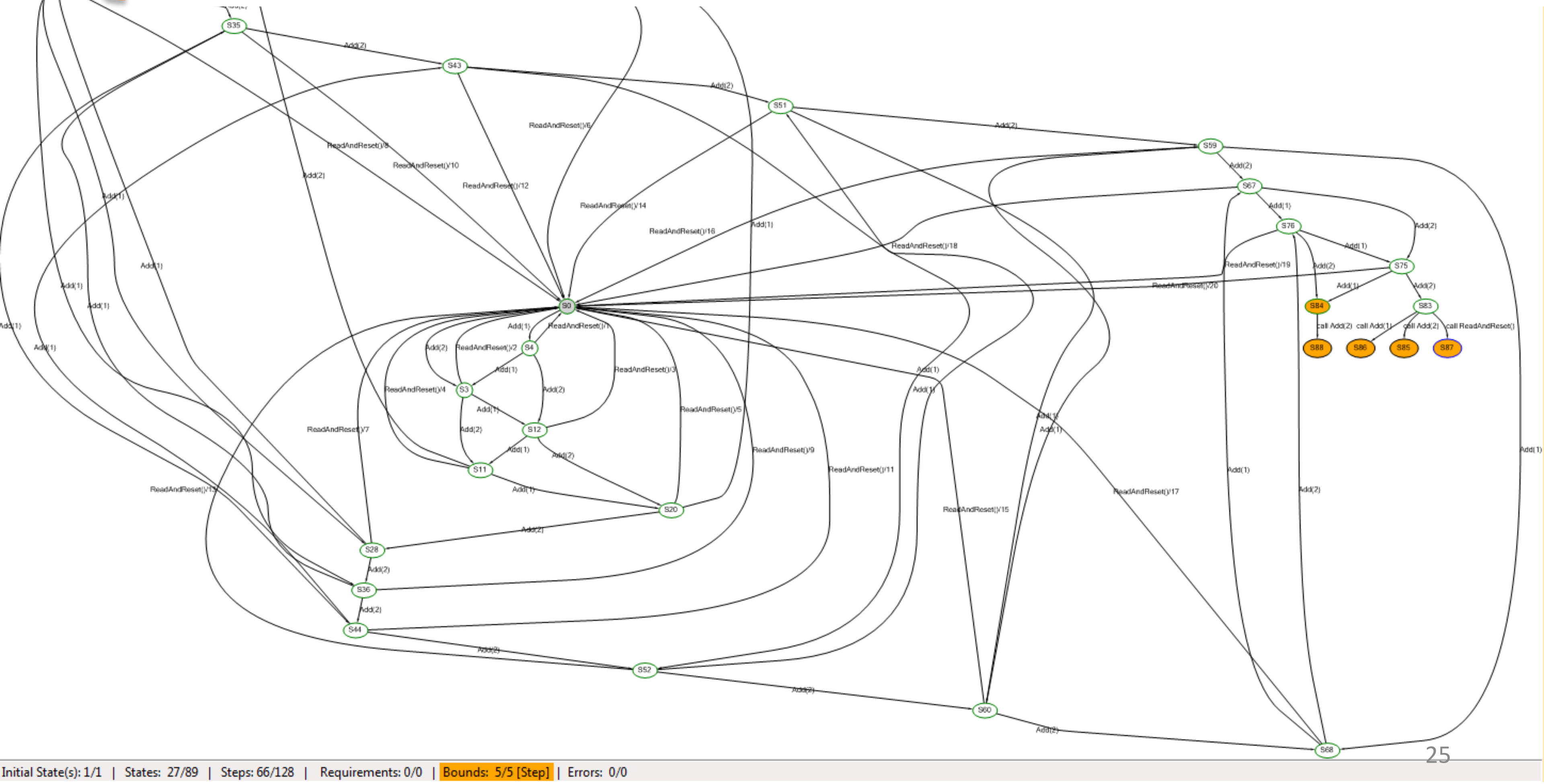
Machine(s) Selected: 1

Machine	Test Enabled	Description	Group	Project	Recommended Views
AccumulatorModelProgram	false			SpecExplorer1	
DoubleAddScenario	false				
SlicedAccumulatorModelProgram	false				
AccumulatorTestSuite	true				

Exploration

1

```
/// Constructs a machine from the model program.
/// Since the model is not finite, this machine explodes
/// and exploration is stopped by a bound.
/// Switch ForExploration makes the machine appear in Exploration Manager.
machine AccumulatorModelProgram() : Main where ForExploration = true
{
    construct model program from ParameterCombination
    where scope = "SpecExplorer1.AccumulatorModelProgram" //The value of the namespace switch can be a .Net na
}
```

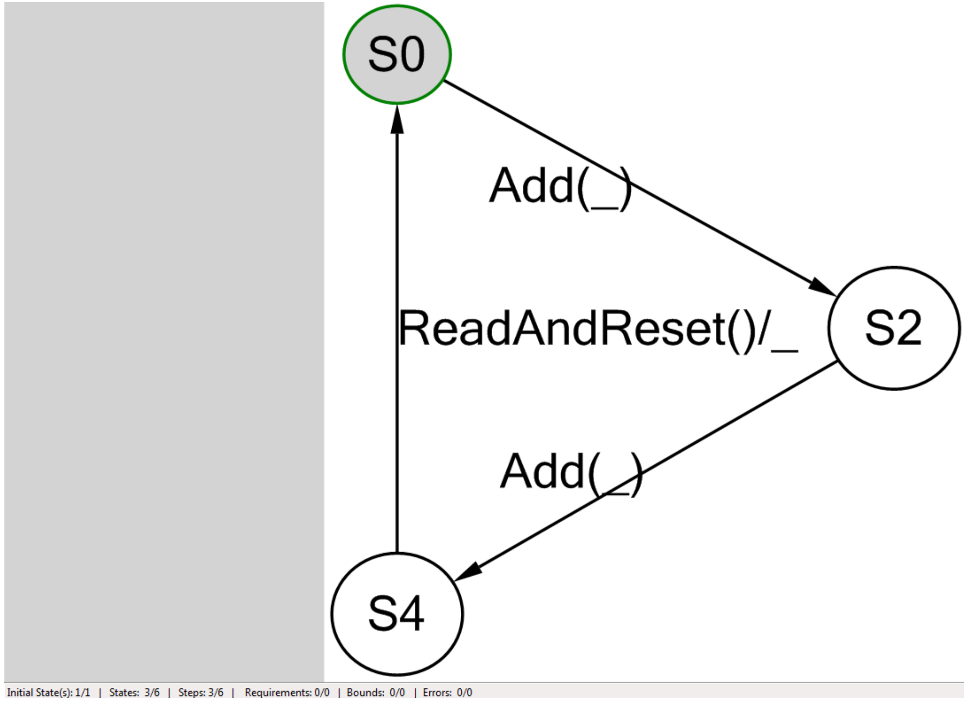


Initial State(s): 1/1 | States: 27/89 | Steps: 66/128 | Requirements: 0/0 | Bounds: 5/5 [Step] | Errors: 0/0

Scenario

2

```
/// Defines a scenario for slicing.
/// When explored on its own, this machine
/// leaves all its parameters unexpanded.
machine DoubleAddScenario() : Main where ForExploration = true
{
    //Omitting the parenthesis for an action invocation
    //is equivalent to setting all its parameters to _ (unknown).
    (Add(_); Add; ReadAndReset)*
}
```

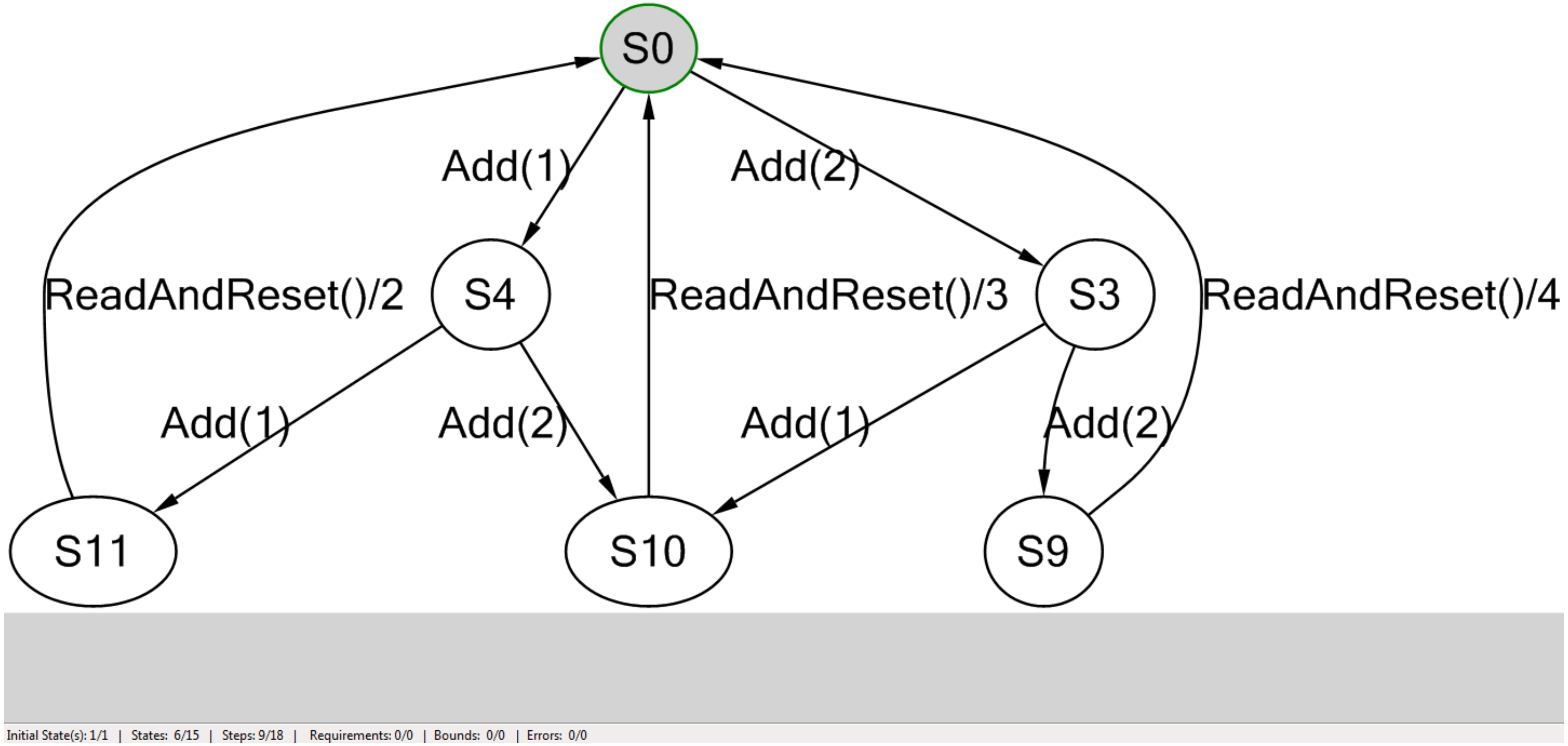


Initial State(s): 1/1 | States: 3/6 | Steps: 3/6 | Requirements: 0/0 | Bounds: 0/0 | Errors: 0/0

Scenario (sliced)

3

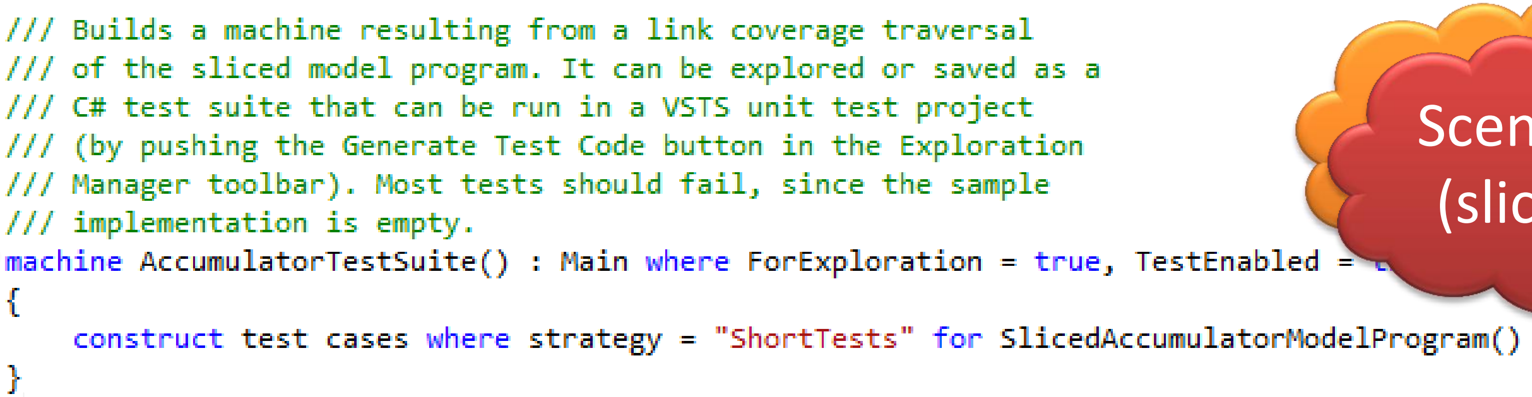
```
/// Selects the slice of the model program
/// that matches the scenario. The model program
/// supplies all parameter and state data omitted from the scenario.
machine SlicedAccumulatorModelProgram() : Main where ForExploration = true
{
    DoubleAddScenario || AccumulatorModelProgram // (synchronized) parallel composition
}
```



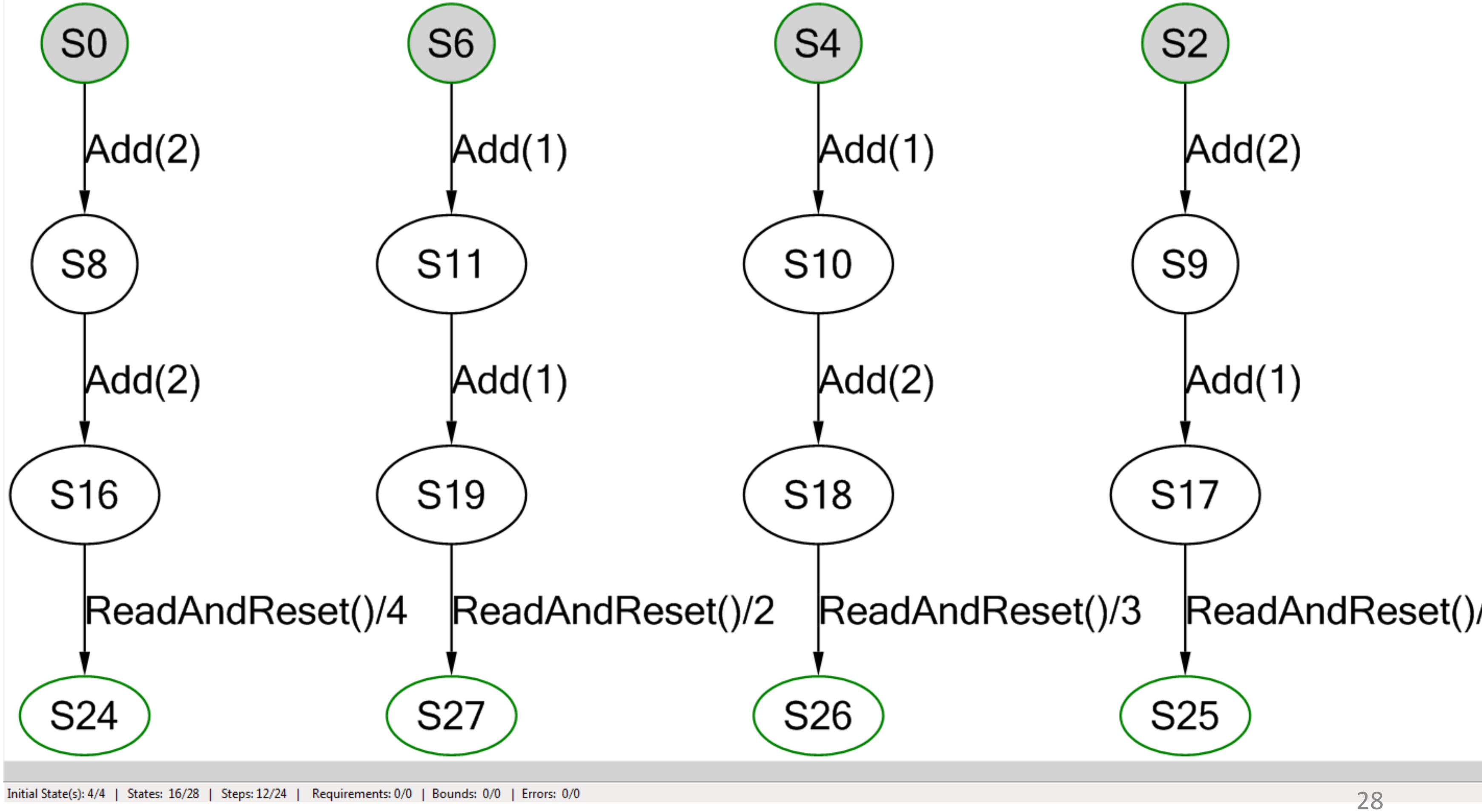
Initial State(s): 1/1 | States: 6/15 | Steps: 9/18 | Requirements: 0/0 | Bounds: 0/0 | Errors: 0/0

Scenario (sliced)

4

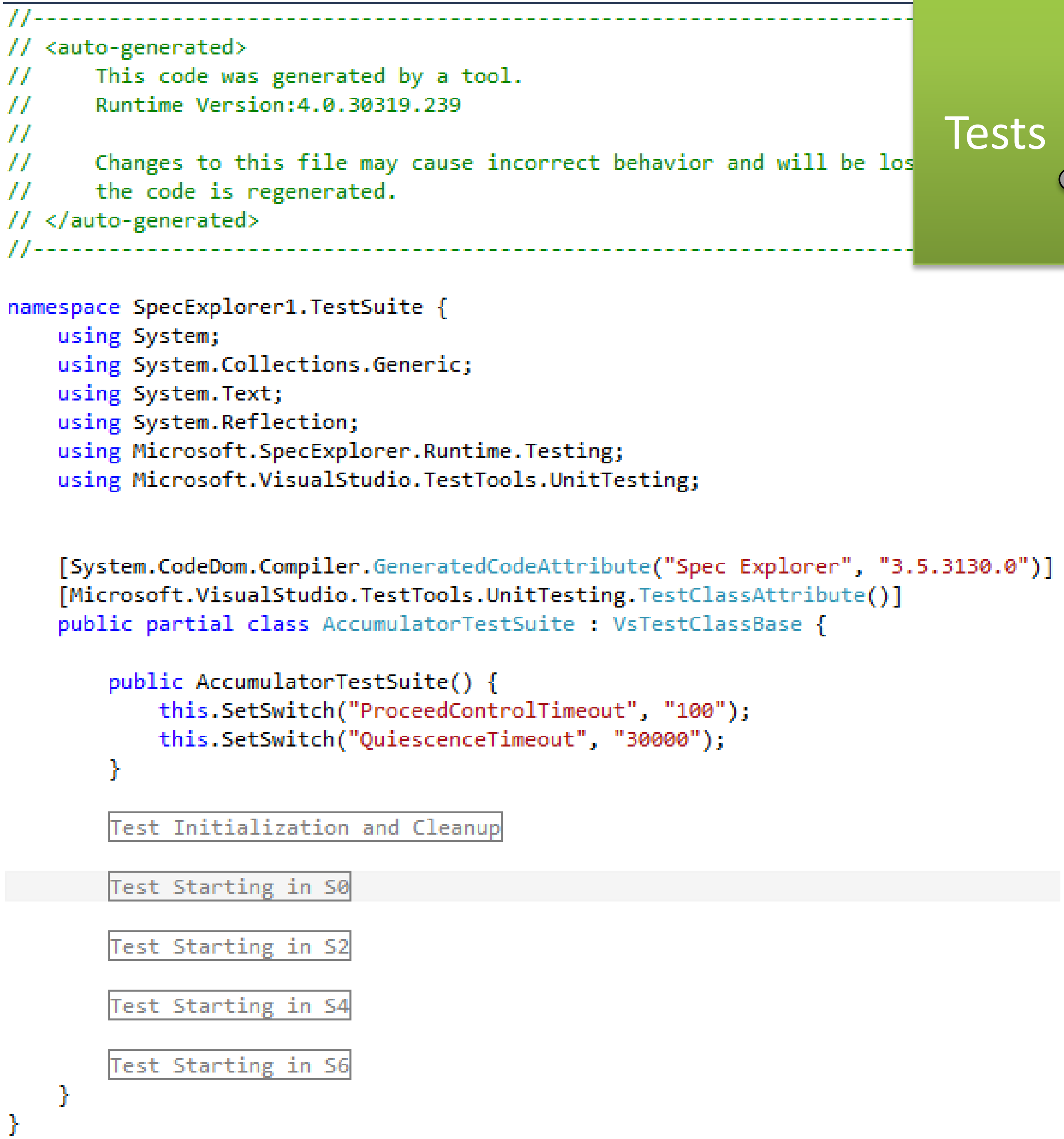


```
/// Builds a machine resulting from a link coverage traversal
/// of the sliced model program. It can be explored or saved as a
/// C# test suite that can be run in a VSTS unit test project
/// (by pushing the Generate Test Code button in the Exploration
/// Manager toolbar). Most tests should fail, since the sample
/// implementation is empty.
machine AccumulatorTestSuite() : Main where ForExploration = true, TestEnabled =
{
    construct test cases where strategy = "ShortTests" for SlicedAccumulatorModelProgram()
}
```



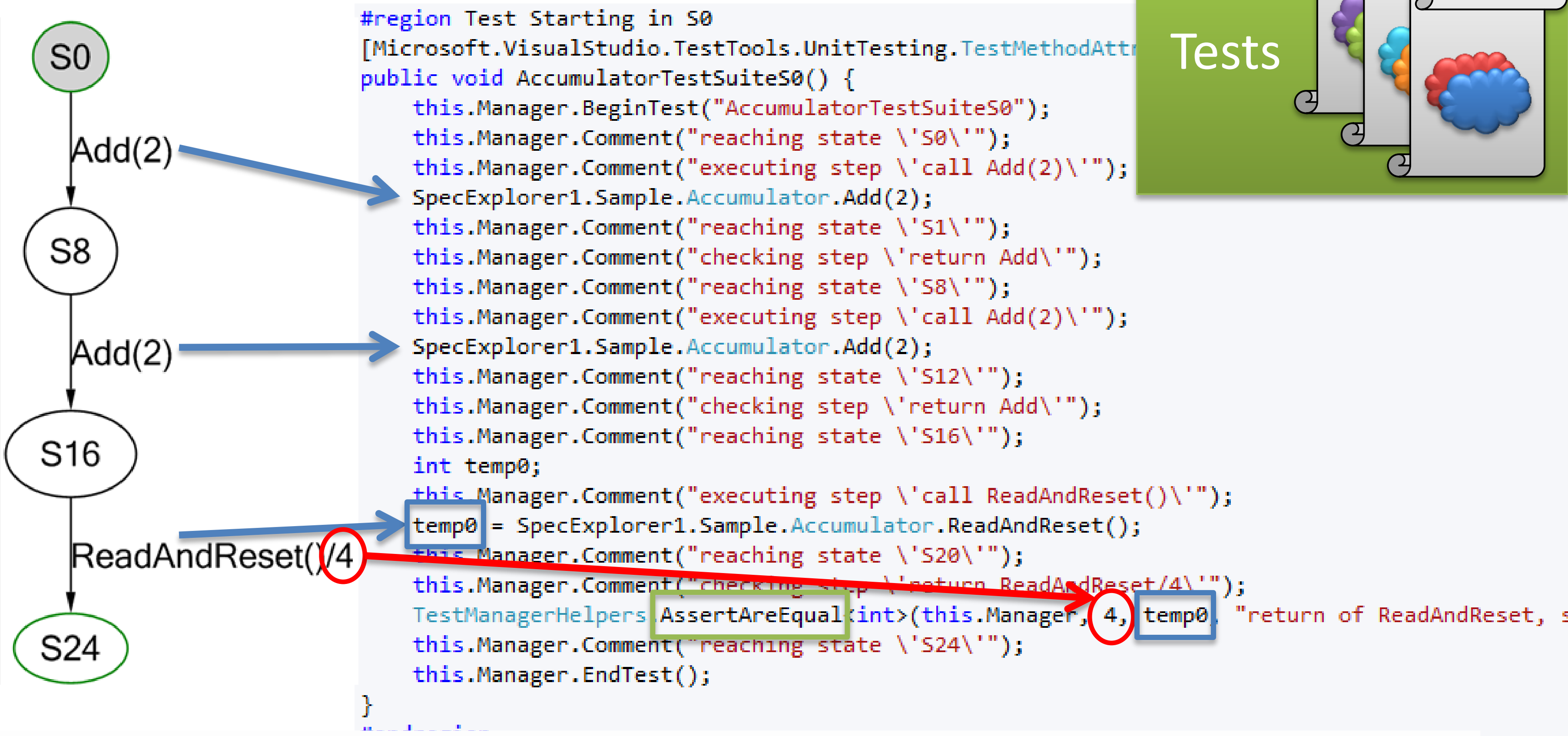
Initial State(s): 4/4 | States: 16/28 | Steps: 12/24 | Requirements: 0/0 | Bounds: 0/0 | Errors: 0/0

Tests

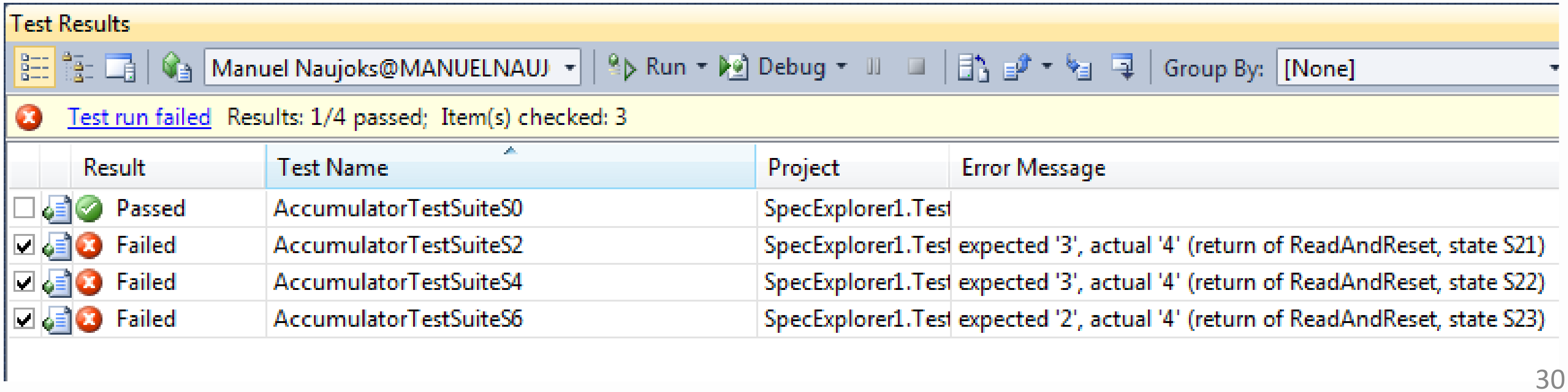


```
//------------------------------------------------------------------------
// <auto-generated>
//     This code was generated by a tool.
//     Runtime Version:4.0.30319.239
//
//     Changes to this file may cause incorrect behavior and will be los
//     the code is regenerated.
// </auto-generated>
//------------------------------------------------------------------------

namespace SpecExplorer1.TestSuite {
    using System;
    using System.Collections.Generic;
    using System.Text;
    using System.Reflection;
    using Microsoft.SpecExplorer.Runtime.Testing;
    using Microsoft.VisualStudio.TestTools.UnitTesting;


    [System.CodeDom.Compiler.GeneratedCodeAttribute("Spec Explorer", "3.5.3130.0")]
    [Microsoft.VisualStudio.TestTools.UnitTesting.TestClassAttribute()]
    public partial class AccumulatorTestSuite : VsTestClassBase {

        public AccumulatorTestSuite() {
            this.SetSwitch("ProceedControlTimeout", "100");
            this.SetSwitch("QuiescenceTimeout", "30000");
        }

        Test Initialization and Cleanup

        Test Starting in S0

        Test Starting in S2

        Test Starting in S4

        Test Starting in S6
    }
}
```

Tests



```
#region Test Starting in S0
[Microsoft.VisualStudio.TestTools.UnitTesting.TestMethodAttr
public void AccumulatorTestSuiteS0() {
    this.Manager.BeginTest("AccumulatorTestSuiteS0");
    this.Manager.Comment("reaching state \'S0\'");
    this.Manager.Comment("executing step \'call Add(2)\'");
    SpecExplorer1.Sample.Accumulator.Add(2);
    this.Manager.Comment("reaching state \'S1\'");
    this.Manager.Comment("checking step \'return Add\'");
    this.Manager.Comment("reaching state \'S8\'");
    this.Manager.Comment("executing step \'call Add(2)\'");
    SpecExplorer1.Sample.Accumulator.Add(2);
    this.Manager.Comment("reaching state \'S12\'");
    this.Manager.Comment("checking step \'return Add\'");
    this.Manager.Comment("reaching state \'S16\'");
    int temp0;
    this.Manager.Comment("executing step \'call ReadAndReset()\'");
    temp0 = SpecExplorer1.Sample.Accumulator.ReadAndReset();
    this.Manager.Comment("reaching state \'S20\'");
    this.Manager.Comment("checking step \'return ReadAndReset/4\'");
    TestManagerHelpers.AssertAreEqual<int>(this.Manager, 4, temp0, "return of ReadAndReset, s
    this.Manager.Comment("reaching state \'S24\'");
    this.Manager.EndTest();
}
#endregion
```

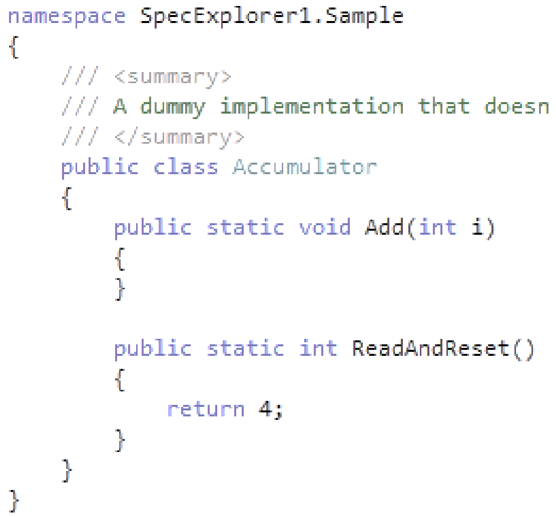


Test run failed Results: 1/4 passed; Item(s) checked: 3

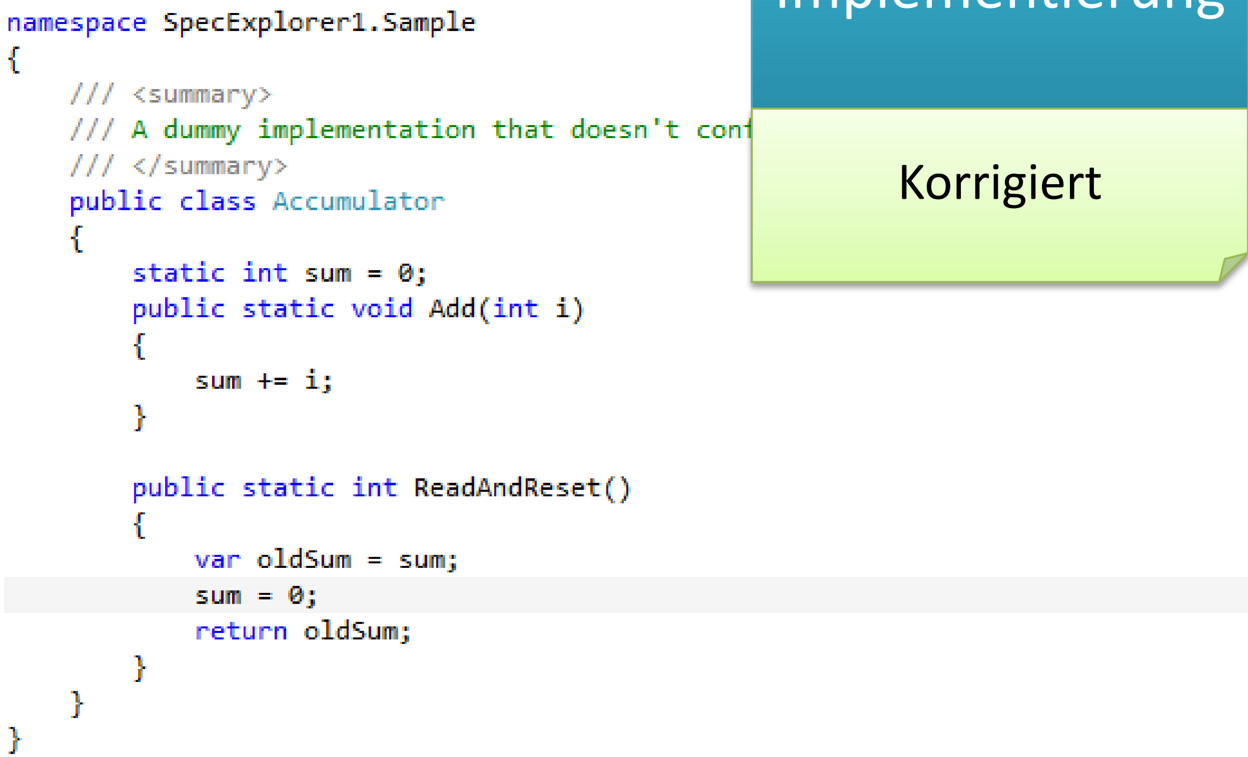
	Result	Test Name	Project	Error Message
☐	Passed	AccumulatorTestSuiteS0	SpecExplorer1.Test	
☑	Failed	AccumulatorTestSuiteS2	SpecExplorer1.Test	expected '3', actual '4' (return of ReadAndReset, state S21)
☑	Failed	AccumulatorTestSuiteS4	SpecExplorer1.Test	expected '3', actual '4' (return of ReadAndReset, state S22)
☑	Failed	AccumulatorTestSuiteS6	SpecExplorer1.Test	expected '2', actual '4' (return of ReadAndReset, state S23)

Implementierung

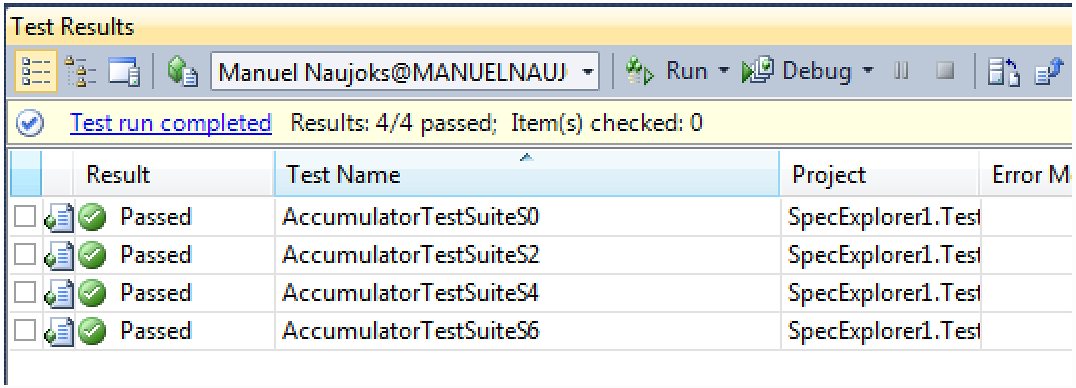
Korrigiert



```
namespace SpecExplorer1.Sample
{
    /// <summary>
    /// A dummy implementation that doesn
    /// </summary>
    public class Accumulator
    {
        public static void Add(int i)
        {
        }

        public static int ReadAndReset()
        {
            return 4;
        }
    }
}
```



```
namespace SpecExplorer1.Sample
{
    /// <summary>
    /// A dummy implementation that doesn't con
    /// </summary>
    public class Accumulator
    {
        static int sum = 0;
        public static void Add(int i)
        {
            sum += i;
        }

        public static int ReadAndReset()
        {
            var oldSum = sum;
            sum = 0;
            return oldSum;
        }
    }
}
```

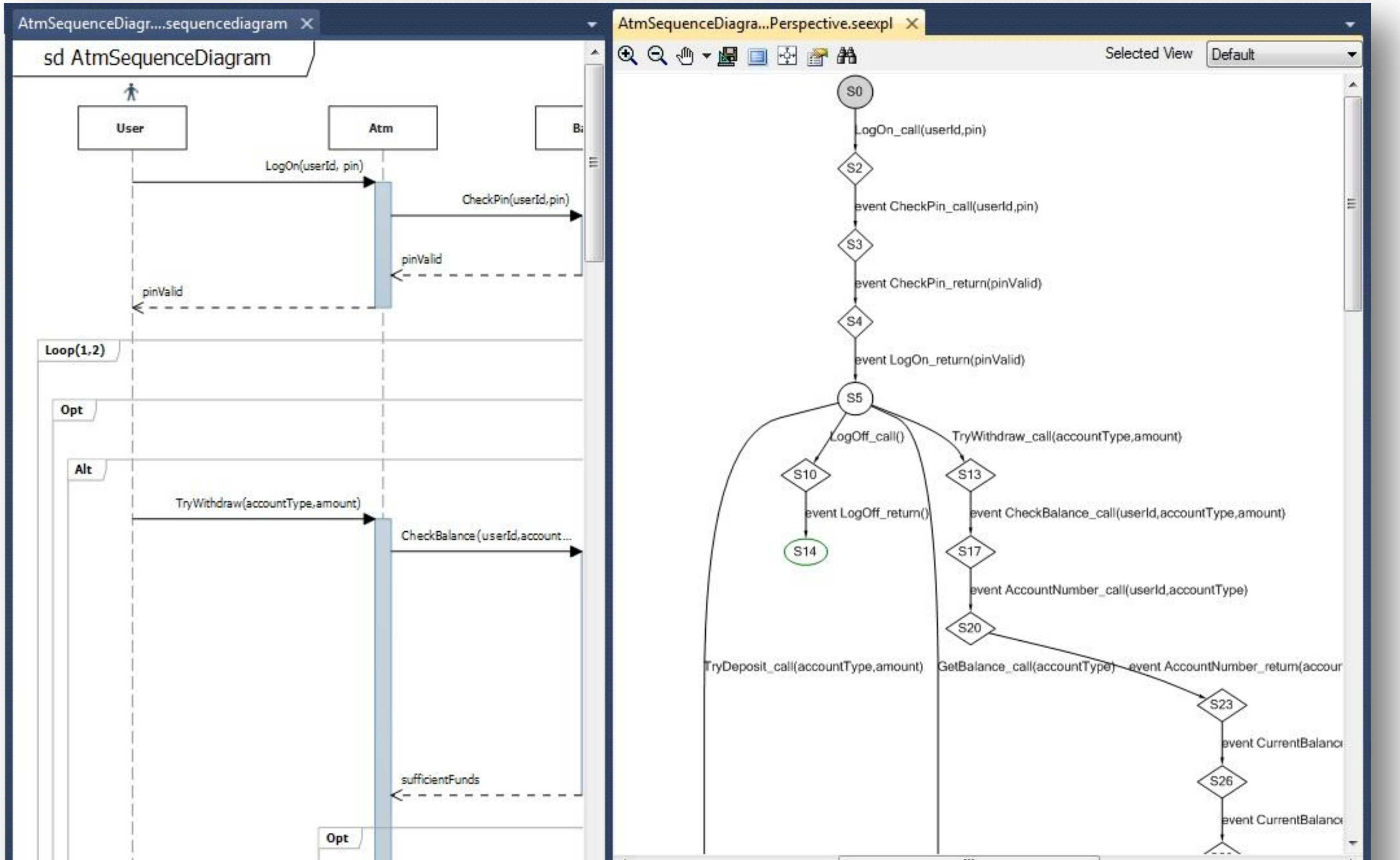


Test run completed Results: 4/4 passed; Item(s) checked: 0

	Result	Test Name	Project	Error M
□	✓ Passed	AccumulatorTestSuiteS0	SpecExplorer1.Test	
□	✓ Passed	AccumulatorTestSuiteS2	SpecExplorer1.Test	
□	✓ Passed	AccumulatorTestSuiteS4	SpecExplorer1.Test	
□	✓ Passed	AccumulatorTestSuiteS6	SpecExplorer1.Test	

UML Extensions

- Szenarien als UML-Sequenzdiagramme


AtmSequenceDiagr....sequencediagram
sd AtmSequenceDiagram
User
Atm
Ba
LogOn(userId, pin)
CheckPin(userId,pin)
pinValid
pinValid
Loop(1,2)
Opt
Alt
TryWithdraw(accountType,amount)
CheckBalance(userId,account...
sufficientFunds
Opt
AtmSequenceDiagra...Perspective.seexpl
Selected View
Default
S0
LogOn_call(userId,pin)
S2
event CheckPin_call(userId,pin)
S3
event CheckPin_return(pinValid)
S4
event LogOn_return(pinValid)
S5
LogOff_call()
TryWithdraw_call(accountType,amount)
S10
S13
event LogOff_return()
event CheckBalance_call(userId,accountType,amount)
S14
S17
event AccountNumber_call(userId,accountType)
S20
TryDeposit_call(accountType,amount)
GetBalance_call(accountType)
event AccountNumber_return(accour
S23
event CurrentBalanc
S26
event CurrentBalanc

Spec Explorer

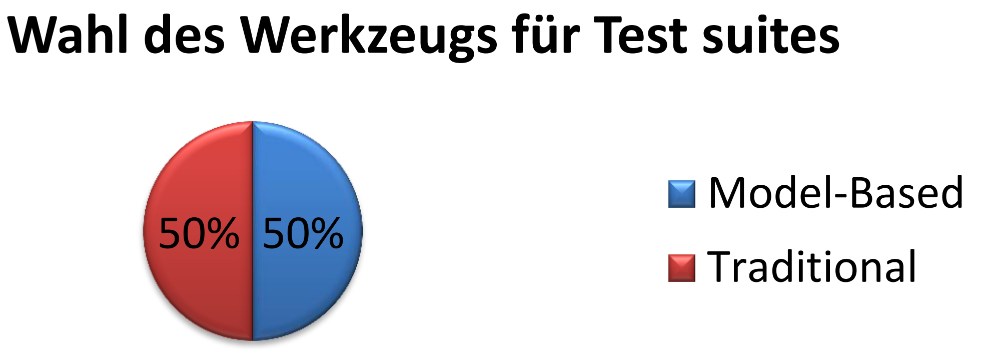
Einsatz bei Microsoft

Einsatz bei Microsoft

- Team seit 2007 in der Windows Server Division
- Development und Testing Team in Beijing
- “Microsoft Open Specifications” Movement
  - http://www.microsoft.com/openspecifications/en/us/default.aspx
  - *Model-Based Quality Assurance of Windows Protocol Documentation*
    - 2008; **Wolfgang Grieskamp**, Nico Kicillof, …
    - http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?tp=&arnumber=4539580&isnumber=4539517?tag=1

Einsatz bei Microsoft

- Anwendung bei Großprojekt
  - – 250 Mannjahre für Testing



Model-Based:

Nach 2,5 Jahren, Effizienzsteigerung von Ø 42%
(Anwendung durch testunerfahrenes Personal)

Spec Explorer

Fazit

Fazit

- Model repräsentiert Zustände der Implementierung
- *State space exploration* erreicht „alle“ Zustände
- Szenarien definierbar
- Szenarien und Exploration „verbinden“
- Automatische Erzeugung von Testfällen für Szenarien
- State-based Testen der Implementierung
- Testfälle müssen bei Modelländerung erzeugt werden

Fazit

- 1-Satz-Zusammenfasssung:

Spec Explorer dient dazu, die Zustände eines Modells mit den Zuständen einer Implementierung zu vergleichen.

Spec Explorer

Ende □

Links

- Spec Explorer Team Blog
  http://blogs.msdn.com/b/specexplorer/
- Spec Explorer 2010 Visual Studio Power Tool
  http://visualstudiogallery.msdn.microsoft.com/271d0904-f178-4ce9-956b-d9bfa4902745
- Spec Explorer 3.5 Reference
  http://msdn.microsoft.com/en-us/library/ee620411.aspx
- UML Extension For Spec Explorer 2010
  http://visualstudiogallery.msdn.microsoft.com/ce73da2a-072f-44d0-ae18-600213b56520
- Using Spec Explorer for Model-Based Test Development
  http://dotnet.sys-con.com/node/163765

Quellen

- The Spec Explorer Story
  http://blogs.msdn.com/b/wrwg/archive/2009/10/28/the-spec-explorer-story.aspx

- What is Model-Based Testing?
  http://blogs.msdn.com/b/specexplorer/archive/2009/10/27/what-is-model-based-testing.aspx

- Abstract State Machines
  http://www.eecs.umich.edu/gasm/

- Abstract State Machine Tutorial
  http://www.di.unipi.it/~boerger/ASMTutorialEtaps.html

- UML Extensions Visual Studio Gallery
  http://visualstudiogallery.msdn.microsoft.com/ce73da2a-072f-44d0-ae18-600213b56520