

# Testing Concurrent Object-Oriented Systems with Spec Explorer

Margus Veanes  
(filling in for Wolfram Schulte)

Foundations of Software Engineering Microsoft  
Research, Redmond

July 20th, 2005

Testing Concurrent OO Systems with Spec Explorer, I-Day,  
Formal Methods 2005, Newcastle, UK

Microsoft  
**Research**

## System testing

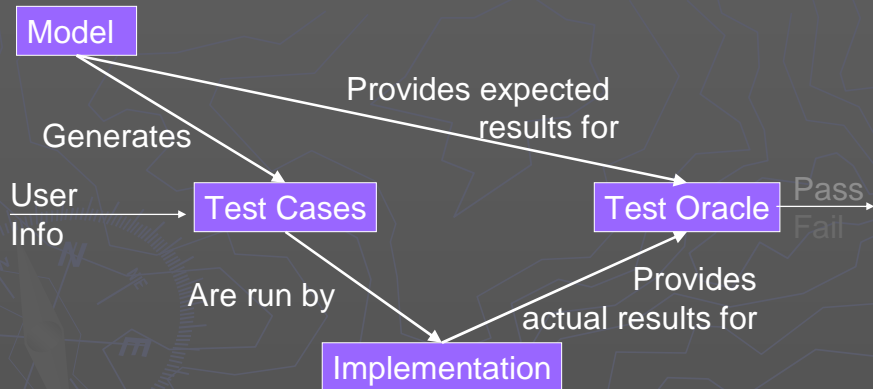
- ▶ Unit testing is insufficient
  - Single unit may work properly in isolation
  - Incorrect interaction between units may cause serious security/reliability failures
- ▶ System-level testing
  - Requires model of system behavior
  - Behavior is often reactive/nondeterministic
    - ▶ Implementation is multi-threaded or distributed
    - ▶ Thread scheduling hard to control
  - State space is typically infinite
    - ▶ Objects, unbounded value types
  - Traditional FSM-based testing is inadequate:
    - ▶ Does not scale

July 20th, 2005

Testing Concurrent OO Systems with Spec Explorer, I-Day,  
Formal Methods 2005, Newcastle, UK

2

# Model-based testing



July 20th, 2005

Testing Concurrent OO Systems with Spec Explorer, I-Day,  
Formal Methods 2005, Newcastle, UK

3

# Our approach

- ▶ Behavior is described by *model programs*
  - Written in *Spec#*
  - Describe reactive system behavior as *interface automata*
  - Use *alternating simulation* for conformance
- ▶ Model exploration, validation and model-based testing are provided by the *Spec Explorer* tool developed at MSR
  - Supports scenario control
  - Provides *offline* as well as *online* testing
  - Views test cases as *game strategies*
  - Checks violations of *alternating simulation*

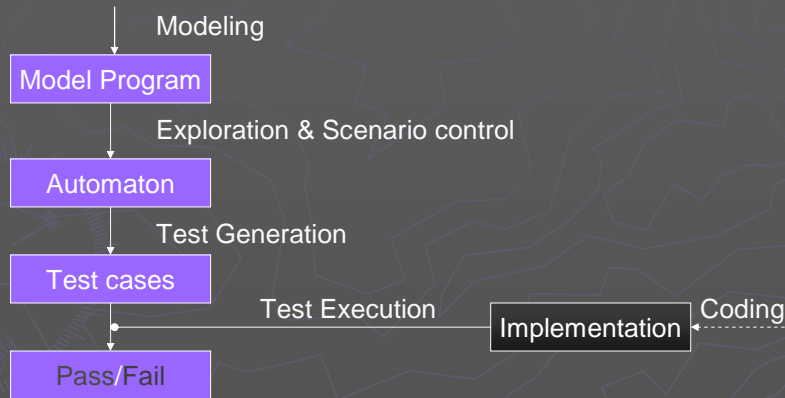
July 20th, 2005

Testing Concurrent OO Systems with Spec Explorer, I-Day,  
Formal Methods 2005, Newcastle, UK

4

# Overview

Spec Explorer supports analysis and conformance testing of concurrent systems with model programs



July 20th, 2005

Testing Concurrent OO Systems with Spec Explorer, I-Day,  
Formal Methods 2005, Newcastle, UK

5

# How it works

- ▶ Modeling
  - Define (infinite) transition system  $T_p$  through program P
- ▶ Exploration & Scenario control
  - Reduce  $T_p$  to a finite test graph G
- ▶ Test generation
  - Generate test cases from G
- ▶ Test execution
  - Run the test cases using the model as the oracle

To avoid state space explosion, exploration, test generation and execution can be combined into a single *online* algorithm

July 20th, 2005

Testing Concurrent OO Systems with Spec Explorer, I-Day,  
Formal Methods 2005, Newcastle, UK

6

# Modeling

- ▶ In Spec#/Spec Explorer:
  - States are mappings of variables to values
    - ▶ ASM states or first-order structures
  - Initial state is given by initial assignment to model variables
  - Actions are defined by method invocations
    - ▶ Actions are either *controllable* or *observable*
  - Preconditions of methods and model invariants define action enabling conditions
  - Transition function is defined by method execution
  - Authoring of models can be done using MS Word

July 20th, 2005

Testing Concurrent OO Systems with Spec Explorer, I-Day,  
Formal Methods 2005, Newcastle, UK

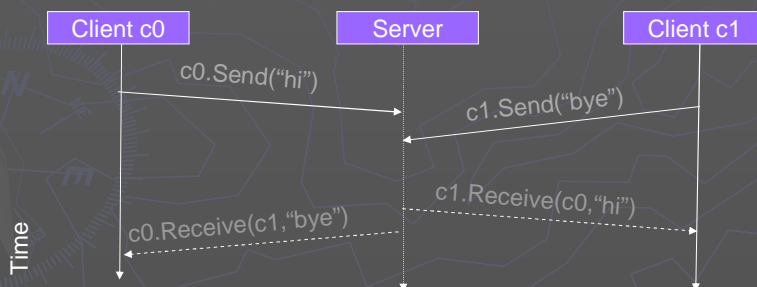
7

## Example: *Chat service*

Service should guarantee FIFO delivery of messages:

Send is a *controllable* action

Receive is an *observable* action



July 20th, 2005

Testing Concurrent OO Systems with Spec Explorer, I-Day,  
Formal Methods 2005, Newcastle, UK

8

# Modeling : Chat system

State:

```
class Client {  
  bool entered = false;  
  Map<Client, Seq<string>> unreceivedMsgs = Map{};  
}
```

Actions:

Controllable:

```
class Client {  
  void Send(string message)  
  requires entered; {  
    foreach (Client c in enumof(Client), c != this, c.entered)  
      c.unreceivedMsgs[this] += Seq{message}; }  
}
```

Observable:

```
class Client {  
  void Receive(Client! sender, string message)  
  requires sender != this &&  
    unreceivedMsgs[sender].Length > 0 &&  
    unreceivedMsgs[sender].Head == message;  
  {  
    unreceivedMsgs[sender] = unreceivedMsgs[sender].Tail;  
  }  
  ...  
}
```

July 20th, 2005

Testing Concurrent OO Systems with Spec Explorer, I-Day,  
Formal Methods 2005, Newcastle, UK

9

# Exploration & Scenario control

- ▶ Exploration is the process of unfolding a rule-based model program into a transition system
  - Actions move the system from state to state
  - State comes from variables containing values
  - Compound data types are possible (sets, maps, sequences, etc...)
  - Objects are identities
- ▶ Scenario control
  - Imposes limits on exploration
    - ▶ Using filters, state groupings, *scenario actions*, ...

July 20th, 2005

Testing Concurrent OO Systems with Spec Explorer, I-Day,  
Formal Methods 2005, Newcastle, UK

10

# Scenario control: Chat system

- ▶ Uses a parameterized scenario action Start.
  - Invoking Start( $n$ ) produces a sequence of  $n$  Create() invocations followed by  $n$  Enter() invocations

```
[Action(Kind=Scenario)]
void Start(int nrOfMembers)
  requires enumof(Client) == Set{};
  {
    Seq<Client!> clients = Seq{};

    // 1-- The given number of
    // clients are created
    for (int i = 0; i < nrOfMembers; i++)
      clients += Seq{Create()};

    // 2-- all clients enter the session
    for (int i = 0; i < nrOfMembers; i++)
      clients[i].Enter();
  }
```

Start(2)

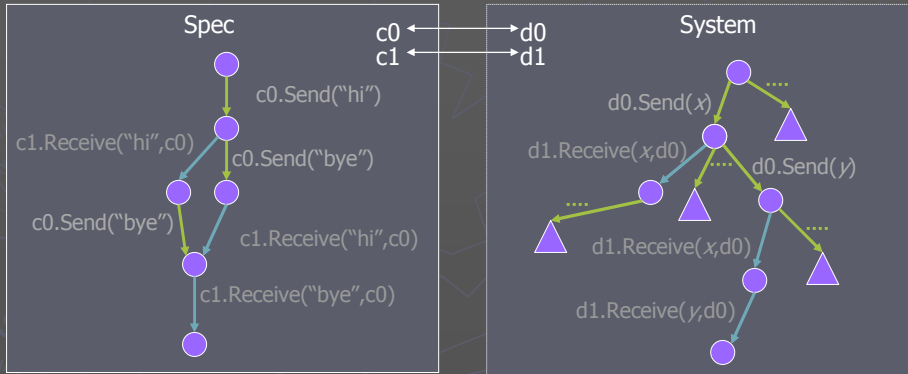
```
Create()/c0;
Create()/c1;
c0.Enter();
c1.Enter();
```

# Test generation

- ▶ The result of exploration is a finite transition system with observable and controllable actions : *an interface automaton A*.
- ▶ Testing can be viewed as a *game* between two players
  - Testing tool (TT)
    - ▶ Making controllable transitions
  - Implementation under test (IUT)
    - ▶ Making observable transitions
- ▶ The "rules of the game" are dictated by *A*
- ▶ Tests are generated from *A*. Test cases are *strategies*
  - A strategy tells what move TT should make in a state where its actions are enabled
  - The purpose of a strategy may be to achieve transition coverage or to reach a certain goal state

# Conformance relation

- ▶ Conformance between the model and the system under test is *alternating simulation with object bindings* [MSR-TR-2005-59]

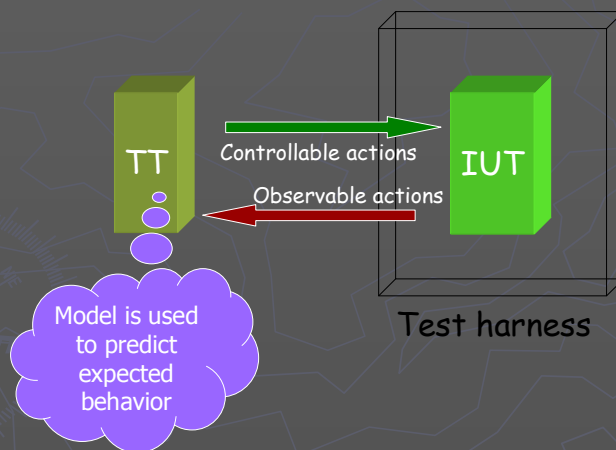


July 20th, 2005

Testing Concurrent OO Systems with Spec Explorer, I-Day, Formal Methods 2005, Newcastle, UK

13

# Testing reactive systems



July 20th, 2005

Testing Concurrent OO Systems with Spec Explorer, I-Day, Formal Methods 2005, Newcastle, UK

14

# Demo: chat

# Experiences

- ▶ Most tool features driven by demands of internal users at MS (mostly testers  $\approx 100$ )
- ▶ Models help discover more bugs during modeling (design bugs) than testing
  - Testers do not get credit for finding those bugs !!!
- ▶ During testing models help discover deep system-level bugs where manual test scenarios don't
  - Such bugs are hard to understand and fix
- ▶ Bugs appear in both models and implementations (the ratio is roughly 50-50)
- ▶ Code coverage is a poor measure for testing concurrent software, often a single execution thread provides the same coverage.
- ▶ New versions of implementation usually require only local changes to models, whereas manual tests often need to be rewritten completely
- ▶ The tool is built on .NET but has been used to test distributed C++ applications. Most test harnesses at MS are built in managed code.

## Some open problems

- ▶ Measuring coverage
- ▶ Failure analysis
- ▶ Repro cases
- ▶ Failure avoidance (during online testing)

Some of these problems can be analyzed in terms of strategy generation

## Recent publications at FSE related to model-based testing

- ▶ Generating finite state machines from abstract state machines, Grieskamp, Gurevich, Schulte, Veanes, *ISSTA'2002*
- ▶ Optimal strategies for testing nondeterministic systems, Nachmanson, Veanes, Schulte, Tillmann, Grieskamp, *ISSTA'2004*
- ▶ State exploration with multiple state groupings, Campbell, Veanes, *ASM 2005*
- ▶ Multiplexing of partially ordered events, Campbell, Veanes, Huo, Petrenko, *TestCom 2005*
- ▶ Play to test, Blass, Gurevich, Nachmanson, Veanes, *FATES'2005*
- ▶ Online testing with model programs, Veanes, Campbell, Schulte, Tillmann, *ESEC/FSE 2005*
- ▶ Model-Based Testing of Object-Oriented Reactive Systems with Spec Explorer, Campbell, Grieskamp, Nachmanson, Schulte, Tillmann, Veanes, *MSR-TR-2005-59*

# Thank you!

- ▶ Visit FSE web site for more information:
  - <http://research.microsoft.com/foundations>

Questions?

July 20th, 2005

Testing Concurrent OO Systems with Spec Explorer, I-Day,  
Formal Methods 2005, Newcastle, UK

19