Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 111 (2005) 5–12
www.elsevier.com/locate/entcs

Model Based Testing in Practice at Microsoft

Keith Stobie[1]

Microsoft Corporation
One Microsoft Way,
Redmond, WA 98052 USA

Abstract

As part of Microsofts Trustworthy Computing [4] initiative the company has sought many ways to increase reliability. One approach being extensively investigated and used is Model Based Testing. With a Finite State Machine modeling tool (TMT) successfully deployed and in use by many test groups, a need for more powerful and flexible modeling has arisen. Several product groups are exploring the use of the Abstract State Machine Language (AsmL) and its associated test tool (AsmL/T). Results from both approaches have shown an increased ability to find defects earlier, including in the specification and design stages, as well as achieve higher structural code coverage on the actual systems under test.

Keywords: formal testing methods, test selection, automated test case generation.

1 Introduction

Most test engineers test software by observing its external behavior, a process called black box testing. In the past most tests have been designed by hand. As the complexity of the software and environment in which Microsoft products must operate increases, demand for designing and especially maintaining tests more easily has arisen. While many testers implicitly or explicitly use models in their heads and some even write them down, the lack of automation prevents fully exploiting the Model Based Testing approach.

A project within the Internet Explorer (IE) team produced a Finite State Machine (FSM) based modeling approach which is now widely deployed. Foundations of Software Engineering group (FSE) in Microsoft Research (MSR)

[1] Email: kstobie@microsoft.com

doi:10.1016/j.entcs.2004.12.004

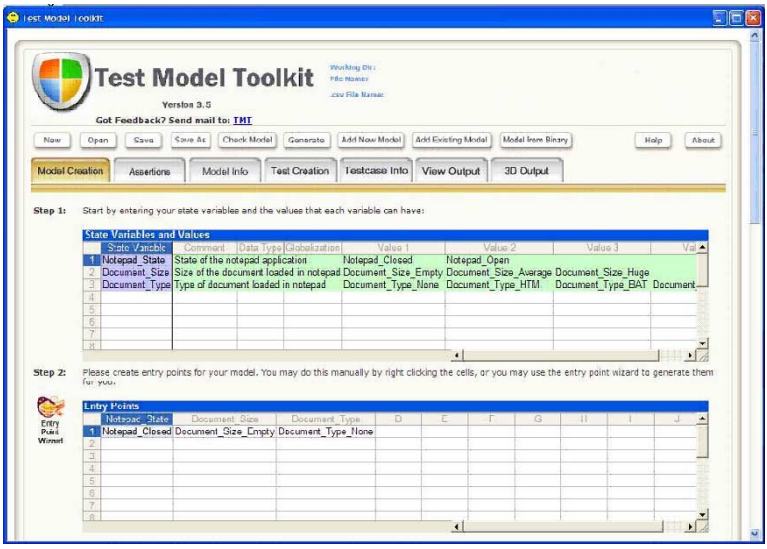

Fig. 1.

created the Abstract State Machine Language (ASML) and showed it off to various product groups. Test groups have been most responsive to using this more advanced approach when FSMs are insufficient.

The communications pillar of Longhorn is called Indigo, which is a set of .NET technologies for building and running connected systems. The Indigo test team has been working with FSE in exploiting AsmL and AsmL/T for testing Indigo as it evolves. Many other teams within Microsoft have also dabbled with AsmL and AsmL/T.

2 Test Model Toolkit

The Test Model Toolkit is a Finite State Machine based program where testers use state tables to model their product. A state table as used by TMT contains three columns: Start State, Action, End State. TMT and the FSM method can be taught to testers with no prior programming experience.

2.1 Generating test cases with test queries

Test queries are used to generate test cases from the model. Queries are expressions of the questions you want to ask your model, and each can be used to generate a large set of test cases. TMT currently exposes six query types:

Exploratory generates random test cases for as many steps as you specify.

Directed generates all available test cases up to a certain number of steps. **Shortest Path** allows you to specify where you want your test case to begin and end, and returns the shortest path between these two points. **All Shortest Paths** finds all the possible ways to begin and end a test case, and returns the shortest paths between all of these points. **ShortestPath-Random** or **ShortestPath-Directed** These queries let you combine two queries. You pick a specific state you would like to get to using a shortest path and then the query creates a random or directed set of test cases from the specific state your picked. You can use these queries to generate test cases that exercise the riskiest areas of your product.

2.2 Assertions

TMT supports simple model checking in the form of assertions. Assertions are statements that the tester makes about their area that should never be broken. Assertions are not only useful for ensuring the quality of the model; but can also be used to ensure the quality of the spec the model was built from.

2.3 Automation

TMT outputs test cases as XML files called XML Test Cases (xtc). TMT also generates an automation template. The tester writes the automation that will run them. This was the first tool that truly helped with test design automation instead of just test execution automation.

2.4 Impact

TMT is used by over 20 teams at Microsoft. Time to automate fell by as much as 88%. BizTalk took one week to generate a set of test cases that took eight weeks by hand. Code coverage increased by as much as 50%. One subteam in Shell in two weeks increased code coverage from 20% to 75%, while the number of test cases increased from about 75 to 2000. Many spec and implementation bugs are uncovered in the course of model creation. [10]

TMT won the testing best practice award inside Microsoft in the spring of 2001.

3 AsmL

The FSE team helped pilot AsmL in many different teams in Microsoft with several resulting papers some of which appear on their web site. [1].

3.1 What is AsmL?

AsmL is an executable specification language based on the theory of Abstract State Machines. The current version is embedded into Microsoft Word and Microsoft Visual Studio.NET. It uses XML and Word for literate specifications. It is fully interoperable with other .NET languages. AsmL generates .NET assemblies which can be executed from the command line, linked with other .NET assemblies, or packaged as COM components. AsmL specifications provide a precise, non-ambiguous way to specify a computer system, either software or hardware. Program managers, developers, and testers can all use an AsmL specification to achieve a single, unified understanding. One of the greatest benefits of an AsmL specification is that you can execute it. That means it is useful before you commit yourself to coding the entire system. By exploring your design, you can answer the following questions:

- Does it do everything you intended it to?
- How do the features interact?
- Are there any unintended behaviors?

3.2 Adoption

Few testers successfully pick up AsmL all on their own (but there are some stellar examples). Indigo testers were provided with a two day training course each year to get them started. Many, but not all, students quickly embraced the concept and attempted usage. However the cost of developing a model is high, and most industry practices do not adequately measure prevented bugs or improved quality. Consequently, testers struggle to get credit for preventing bugs by asking clarifying questions and removing specification bugs long before executable code or test cases are ever developed.

3.3 Impact

Indigo is based on a number of industry defined standards for Web Services (WS). During the drafting of these standards parts of the Indigo test team modeled the specifications with help from FSE. Modeling the WS-Routing specification [6] in about a week we found twenty issues using the AsmL model which had taken months for authors to find out the same in design / architecture meetings. Modeling the WS-Policy specification [7] found six issues which all got fixed. We found additional issues in revisions of the specification. Modeling the WS Atomic Transactions specification [10] numerous specification issues were raised. New Passport login protocol modeled in AsmL raised so many specification issues that the specification was rewritten entirely.

4 AsmL/T

AsmL gave rise to the AsmL Test Tool [2]. Using the AsmL Test Tool, you can generate tests directly from an AsmL model. Both parameter generation and test sequence generation are supported. The tool also allows you to run your AsmL model in parallel with the actual implementation in order to check the implementation's conformance to its specification. The tool exposes a simple automation interface so it can easily be embedded into your test harness.

You can tell in this first generation tool that it was created as a collection of Researchers areas because the same information has to be provided in multiple ways for items such as Methods and Actions.

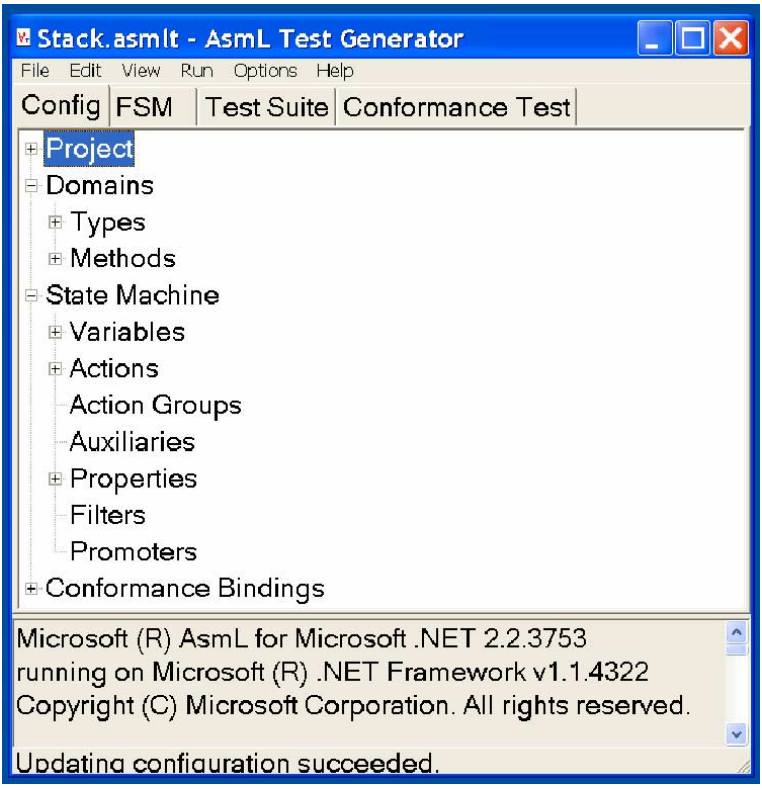

Fig. 2.

4.1 Adoption

AsmL/T allowed testers to better leverage the effort they put into creating models. Now, not only were specifications better, but test cases could be

generated and an oracle provided. Still the cost was high as this was basically a research prototype. Documentation was frequently missing, concepts were stated in unfamiliar (research) terms, and there was a steep learning curve. Only because of training, consulting, mentoring, and coaching from the FSE team was adoption generally possible.

4.2 Impact

In WS-Routing testing of Web Service Extension 1.0 [8], one incompatibility issue was found after it had already been traditionally tested. In an early implementation of WS-Policy by Indigo, AsmL/T parameter generation was used to generate variations of policy which found 13 bugs in the implementation. More than three very insidious bugs were found in the Indigo early implementation of WS-AtomicTransaction.

5 Model Based Testing : Research to Practice

Researchers enjoy tackling the tough unsolved problems. Unfortunately it is frequently the simple solved problems that provide a stumbling block for practitioners to incorporate the advances into their daily practice. Real testers run fully automated regressions without any GUI or prompts. Real testers have complete control to reproduce any scenario, archive it, and reduce it into a minimal case [5] as part of a bug report for development.

6 Futures

Microsoft hopes to promote Model Based Testing as a best practice across the industry and many practicing testers within Microsoft speak at industry conferences and write for industry magazines and webzines on the benefits of MBT and how to easily get started. [11]

TMT is being evolved into an Integrated Test Environment that includes an enhanced version of the FSM Modeling, N-wise parameters combinations, and allied test technologies.

The FSE group has taken Indigo Tests feedback and is creating a next generation of the AsmL and testing tools that are more intuitive (shorter learning curves and closer to current tester experience), easier to debug, and well tested and supported.

7 Futures

Finite State Machines and other simple models like Cause Effect Graphing should already be in use as a best practice for Model Based Testing across the industry, but are not. The tools are still perceived as too expensive or too complex by most testers. Constant evangelism with case studies and consulting are required to convince most test groups still. Further, when industrial testers finally embrace MBT, they frequently find the simple models are not rich enough to handle their problems. Providing simple ways for testers to expand from dipping their toes in the water with simple models to handling great swaths of their testing domain using advanced models that also allow model checking and execution are required.

Acknowledgements

This work is due to a wide variety of people including everyone in the Foundations of Software Engineering (FSE) group within Microsoft Research (which has included Colin Campbell, Yuri Gurevich, Wolfgang Grieskamp, Wolfram Schulte, Nikolai Tillmann, and Margus Veanes) and many testers within the Indigo test team, especially Sara Wong.

References

[1] Abstract State Machine Language, Microsoft Research.
URL http://research.microsoft.com/fse/asml/

[2] The AsmL Test Generator tool, Microsoft Research.
URL http://research.microsoft.com/fse/AsmL/doc/AsmLTester.html

[3] M. Corning, " Confessions of a Modeling Bigot: Parts III-IV " ASPToday, Dec02 & Jan03.
URL http://www.asptoday.com/content.asp?id=2043

[4] *Trustworthy Computing : Reliability*
URL http://www.microsoft.com/mscorp/twc/reliability/default.mspx

[5] A. Zeller and R. Hildebrandt, *Simplifying and Isolating Failure-Inducing Input.*, IEEE Transactions on Software Engineering **28(2)**, February 2002, pp. 183-200

[6] H.F. Nielsen, S. Thatte, *Web Services Routing Protocol (WS-Routing)*, October 23, 2001
URL http:///msdn.microsoft.com/library/default.asp

[7] D. Box, et al, *Web Services Policy Framework (WSPolicy)*, 28 May 2003
URL http://www.ibm.com/developerworks/library/ws-policy
URL http://msdn.microsoft.com/ws/2002/12/Policy/

[8] *Web Services Enhancements (WSE) 1.0 SP1 for Microsoft .NET*
http://www.microsoft.com/downloads/details.aspx

[9] L.F. Cabrera, et al, *Web Services Atomic Transaction (WS-AtomicTransaction)*, September 2003.
ftp://www6.software.ibm.com/software/developer/library/ws-atomictransaction.pdf
URL http://msdn.microsoft.com/library/en-us/dnglobspec/html/wsat.asp

[10] J. Taylor, *Testing Internet Explorer: From Underdog To Best-In-Class Browser*, StarWest 2002.

[11] H. Robinson, *Finite State Model-Based Testing on a Shoestring*, StarWest 1999
URL http://www.model-based-testing.org/shoestring.htm