Model-Based Testing in Industry

Keith Stobie
Doyenz
Work while at Microsoft (Engineering Excellence)


State
Space
Training
Maintenance
Support

Outline

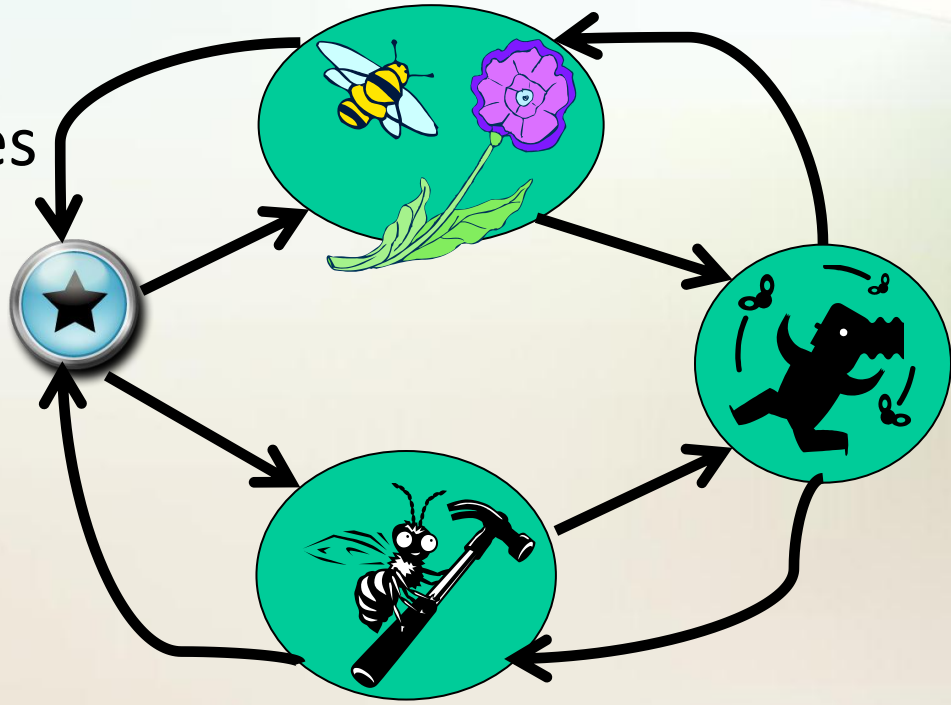
model-based testing: umbrella of approaches that generate tests from models [ETSI ES 202 951 V1.1.1 (2011-07)]

- Types of modeling
- Slanted History of Microsoft modeling
- Requirements for sustained MBT
- Microsoft usage today: API, UI, Protocol, Web

(system) model: computer-readable behavioral model that describes the intended external operational characteristics of a system [ETSI]

Model-Based Testing

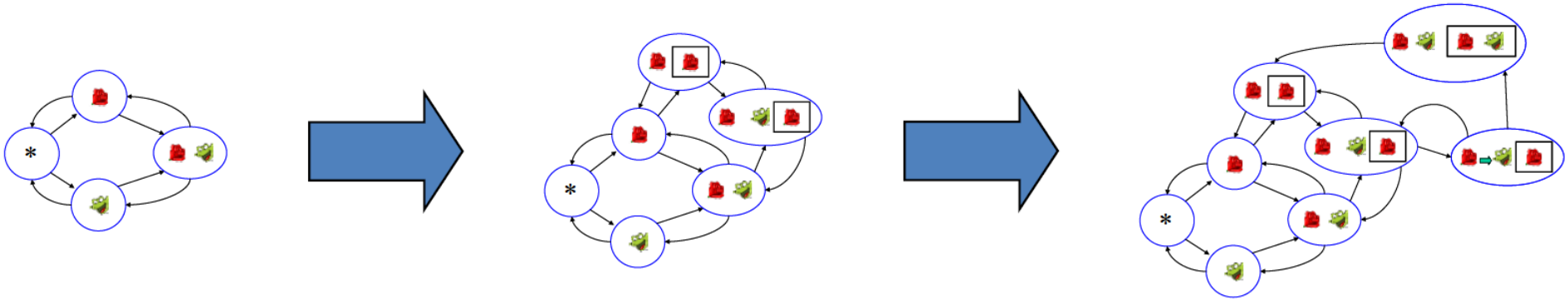
- Automating rules of thumb
- Test Data generation (combinatorics & grammars)
- Finite State Machines
- Abstract State Machines / Action Machines
- Petri Nets
- Performance / etc.



Test Generation
vs. Model Exploration

Models Supporting Automated Exploratory Testing

- Automating Rules of Thumb
- You can start with a small model and build in the direction you think will do the most good for you.



Harry Robinson - Exploratory Test Automation
http://www.harryrobinson.net/ExploratoryTestAutomation-CAST.pdf

Microsoft MBT History

2000 : IE uses Test Model Toolkit (TMT)

2001 : Indigo uses Asml/t (Abstract State Machine Language/test)

2003 : Temporal Logic of Actions (TLA) used

2004 : MS Research Spec Explorer with Spec#

2004 : Model Design Environment (MDE)

2005 : Kokomo & PICT & PEX

2007 : Spec Explorer for **V**isual **S**tudio / nModel

2010 : Spec Explorer VS PowerTool

Others: Goldilocks, Petrinetor, Rapid Api Test Tool (RATT)

Pairwise: PrioGen

Test Model Toolkit

TMT

Step 1: Start by entering your state variables and the values that each variable can have:

State Variables and Values

	State Variable	Value 1	Value 2	H	I	J
1	calculator	calc_off	calc_on			
2	window	standard	scientific			
3	about	about_off	about_on			
4	display	cleared	number			

Step 2: Please create entry points for your model. You may do this manually by right clicki
you may use the entry point wizard to generate them for you.

Entry Point Wizard

Entry Points

	calculator	window	about	display	E	F	
1	calc_off	standard	about_off	cleared			

Finite State Machines

- Easy to grasp and learn, thus easily adopted. In 2001 TMT had over 20 teams using it.
- Doesn't handle complexity of most real systems
- TMT and its successor MDE are on the wane at Microsoft.

+ TMT had Great Community support

- TMT Unable to scale. State explosion.

Asml Topological sort

```
class Vertex
var V as Set of Vertex
var E as Set of (Vertex,Vertex)
var S as Seq of Vertex = []

topsort()
  step while ToSet(S) ne V
    let X = V - ToSet(S)
    let Y = {v | v in X where
              not(exists u in X where (u,v) in E)}
    S := S + [(any v | v in Y)]
```

TLA for WS-Atomic Transaction

The TC ends the volatile prepare and begins the durable prepare. It does this when it has received *Prepared* or *ReadOnly* messages from every participant that registered as volatile, and it sends a *Prepare* message to every participant that registered as durable.

$$
\begin{aligned}
&\land\ tcData.st = \text{“preparingVolatile”} \\
&\land\ \forall\, p \in Participant : tcData.reg[p] \neq \text{“volatile”} \\
&\land\ tcData' = [tcData \ \text{EXCEPT}\ !.st = \text{“preparingDurable”}] \\
&\land\ msgs' = msgs \cup [type : \{\text{“Prepare”}\}, \\
&\qquad\qquad dest : \{p \in Participant : \\
&\qquad\qquad\qquad tcData.reg[p] = \text{“durable”}\}] \\
&\land\ \text{UNCHANGED}\ \ iState
\end{aligned}
$$

TLA WS-Atomic Transaciton

Formal Specification of a Web Services Protocol (2004)

*research.microsoft.com/en-us/um/people/lamport/**tla**/wsfm.pdf*

- questions because original specification not clear enough
  to permit their formal specification.
- Most of the effort resolving these issues
  many discovered only while writing the specification.
- Model checking revealed behaviors unanticipated by the designers.
  → clarifications and changes to the original specification.
- *Colin Campbell found a minor error in our specification when translating it into the AsmL specification language.*

a routine exercise for Lamport, as it would have been for anyone with a moderate amount of experience specifying concurrent systems. Using TLA+ for the first time was a learning experience for Fritz. TLA was a brand new world for product developers, who had never been exposed to formal methods before.

research.microsoft.com/en-us/um/people/lamport/pubs/pubs.html#wsfm-web

Mathematical Languages

- ASML (tried to be Visual Basic like)
  Resistance to new language & sets

  Note new acceptance of LINQ and Lambda expressions

- TLA+ (math like)
  Very difficult to get traction in product groups.

Spec # (by contract)

- Spec Explorer 2004 used Spec#
- Targeted towards specifiers of products (Product Managers [PMs])
- Integrated into MS-Word for literate programming.
- Didn’t catch on with PMs
- Used by test, but resisted because non-standard language
- Like Asml still problems with state explosion and controlling the model

Experience of MBT at Microsoft (2005)

[1] ASML

– More difficult to quantify improvements

[2] General improvements

– 600 of 5000 testers are currently involved in some form of model based testing

– More than 35 Microsoft product teams are engaged in model-based testing efforts

– Microsoft Technical Education is creating several model-based testing courses.

– "Model-based testing will be the primary method of creating test plans and test cases" – Craig Zhou, Director of Microsoft's Windows Test Infrastructure

[1] Test Model Toolkit (TMT)

- Time to automate fell by as much as 88%
- Biztalk took 1 week to generate test that took 8 weeks by hand
- Code coverage increased by as much as 50%
- One team in 2 weeks increased coverage from 20% to 75%, and increased test cases from 75 to 2000
- TMT won Testing Best Practice Award inside Microsoft

1] Model Based Testing in Practice at Microsoft, Keith Stobie, Electronic Notes in Theoretical Computer Science, 111 (2005) 5 – 12. www.elsevier.com/locate/entcs

[2] Obstacles and opportunities for model-based testing in an industrial software environment, Harry Robinson, Microsoft From : http://shakti.it.bond.edu.au/~sand/TAW06/Model%20Based%20Testing%20with%20Action%20Words.pdf

Ten Years of Model Based Testing A Sober Evaluation (2006)

Alan Hartman, IBM *http://react.cs.uni-sb.de/mbt2006/talks/ModelBasedTestingSoberEvaluation.PDF*

1999 – IBM US file system test – Never Used Again

2001 -- GDL Models -- No repeated use of the tools

2003 -- Modeling with spreadsheet - No repeat business

2005 -- reflection to generate models -- tester never sees a model

- Barriers to Model Based Testing
  - Process shift – Up front investment in test
  - Personnel shift – Higher education & sophistication
  - Tools – Still bleeding edge

MBT another Automation

MBT has same requirements as any other test automation

- Return On Investment
- Don’t attempt to automate everything
- Extensible is better
- Plan for maintenance
- Support & Community

Stateless Models 2005 Grammars & Combinatorics

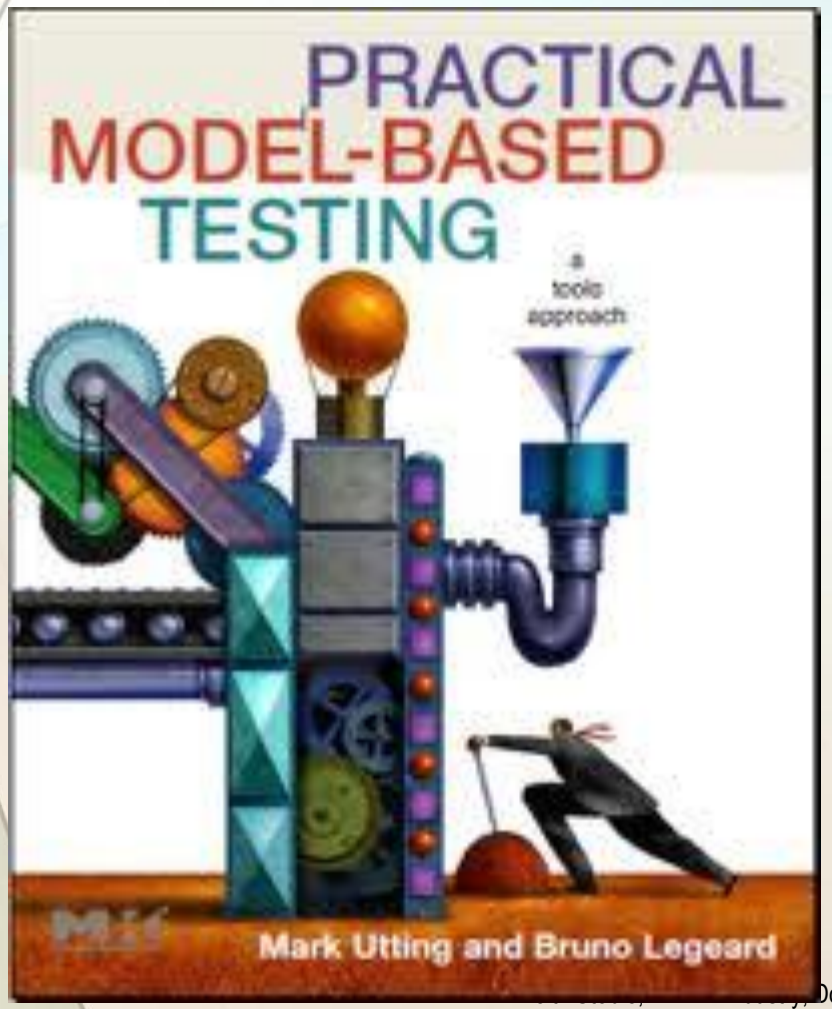
- Pairwise Independent Combinatorial Testing (PICT) is most popular and widely used “modeling” tool at Microsoft

- Input generation based on “model” of inputs. (like AETG, Hexawise)

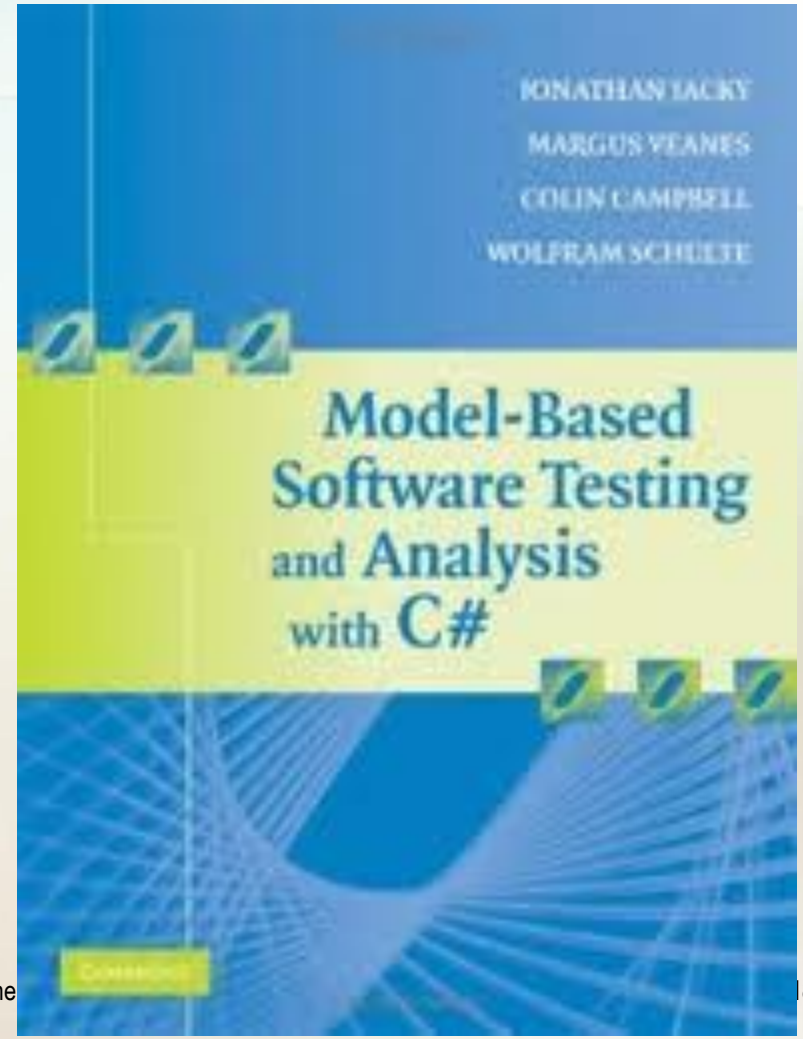
- No oracle (except trivial “legal”/ “ not legal” inputs)

Books

Dec. 2006



Nov. 2007



Spec Explorer 2008 (now 2010)

- Modeling in mainstream languages and environments
  Standard .Net languages (CLR) - C#
- Visualization of exploration and actual generated test suite
- Dealing with state explosion by scenario slicing
- Parameter combinations & combinatoric data control
- **Training and support resources**

  [Nico Kicillof, Microsoft: Model-Based testing with .NET]

  - Public blogs http://appmbt.blogspot.com/
    http://testingwithkunal.com/category/model-based-testing/
  - Forum on MSDN
  - full video series for online training class

C# Model code (ATSvc)

```
static var jobs = new MapContainer<int,JobInfo>();

[Rule]
static bool AddJob(JobInfo info, out int jobId)     {
  jobId = AllocateUniqueId();      jobs[jobId] = info;
  Capture(1, "When adding a job, the server MUST return...");
  return true;
}

[Rule]
static bool DeleteJob([Domain("JobIdsInUse")]int id)   {
  Condition.IsTrue(jobs.ContainsKey(id));
  jobs.Remove(id);   DeallocateId(id);
  return true;
}

[Rule]
static bool GetJobInfo(int id, out JobInfo info)    {
  Condition.IsTrue(jobs.ContainsKey(id));
  Capture(2, "When querying information about a job,… ");
  info = jobs[id];
  return true;
```

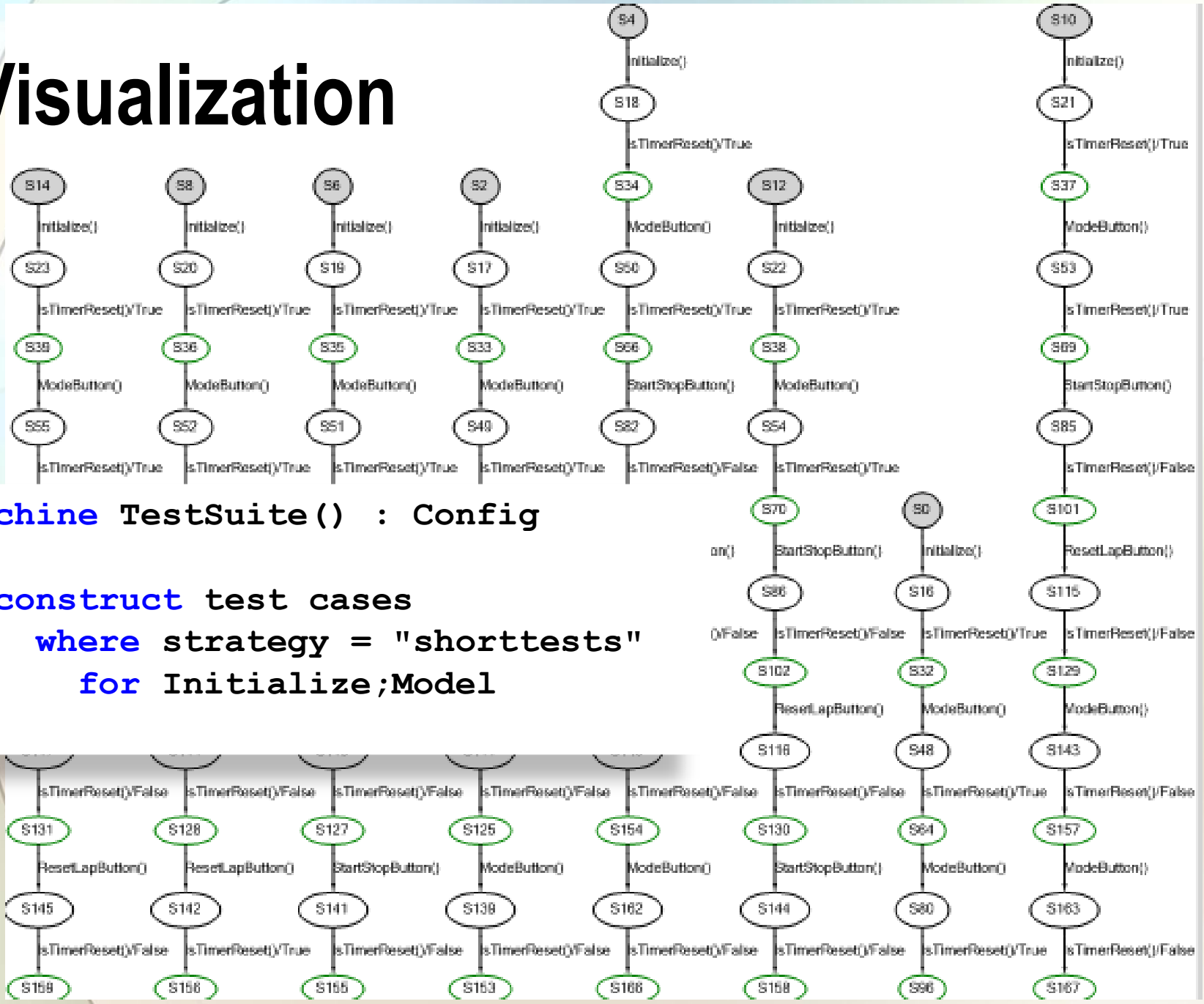
Very general state

Requirements tracking

Domain restriction

Simple Pre-Conditions

Visualization




```
machine TestSuite() : Config
{
  construct test cases
    where strategy = "shorttests"
      for Initialize;Model
}
```

Mastering State Explosion (Slicing)

Define domains. Constrain sequences and states to consider

Model behavior slicing techniques:

- Parameter selection
- State filtering
- Behavioral restriction using trace patterns

Slicing requires human intervention
It can fundamentally reduce test coverage

Smart slicing is an art
Over-slicing is a common problem.

Data & Combinations

- Equivalence Classing Infinite domains
  Must still choose reasonably small number of believable and acceptable values
- 1-wise often good enough
  Cartesian product frequently too thorough

Parameter Selection in Cord

- Add constraints and domains to action definitions

```
config ConfigWithParameters : Config    {
  action bool IATService.AddJob(JobInfo info, out int jobId)
    where
    {.
      Condition.In(info.Command, "x", "y");
      Condition.In(info.Time, 340000, 600000);
      Condition.IsFalse(info.Command == “x”
                        & info.Time < 600000);
    .};
```

- Define parameters in trace patterns

```
machine ModelProgramWithBoundParameters() : Config  {
  bind AddJob(JobInfo(Command={"x","y"},
                Time={340000, 600000}), out _)
  in  ModelProgram
```

Behavioral restriction with trace patterns

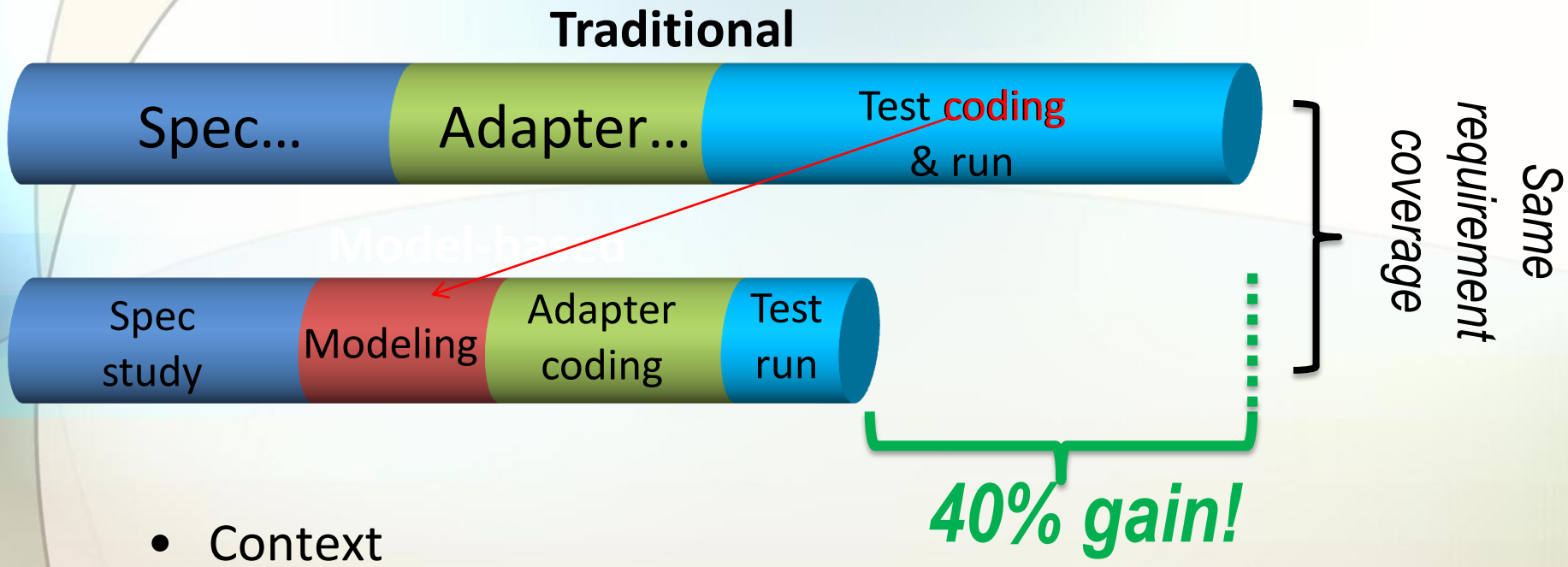
Spec Explorer allows definition of trace patterns

- Regular expressions over actions (and other operators)

Such patterns can be composed with the model

```
machine AddTwoJobsPattern() : Config
{
  AddJob; AddJob; GetJobInfo; DeleteJob;
  AddJob; GetJobInfo; DeleteJob*
}
machine ModelProgramWithTwoJobsPattern() : Config
{
  AddTwoJobsPattern
  || ModelProgramWithConfigParameters
}
```

Protocol modeling



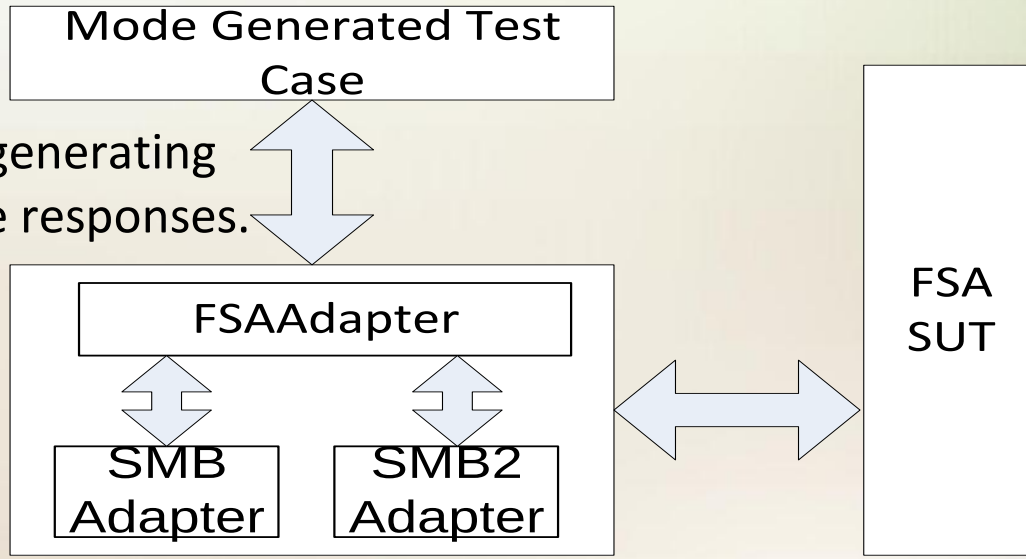
- Context
  - Windows protocol compliance (Web protocols, RPC, others)
  - Total effort: 250 person years (mostly junior vendors)
  - Statistical significance validated by Microsoft Research
- MBT saved Microsoft 50 person years of tester work!

FSA Test Suite Overview

- Test Suite Approach : Model-Based Testing (MBT)
  - **Sequential Operational Flow** e.g. Phase1, Phase2…
  - **Complex combinations of parameters** e.g. 3.1.5.1 "Application Requests an Open of a File", Phase 1 - Parameter Validation, there are 10 parameters mentioned: and the different values of these parameters are combined to determine different outputs.
- Adapters
  - **Protocol Adapter:**

    FSAAdapter has methods for generating the requests and receiving the responses.
  - **Transport Adapter:**

    use SMB or SMB2 as transport.



SMB Example

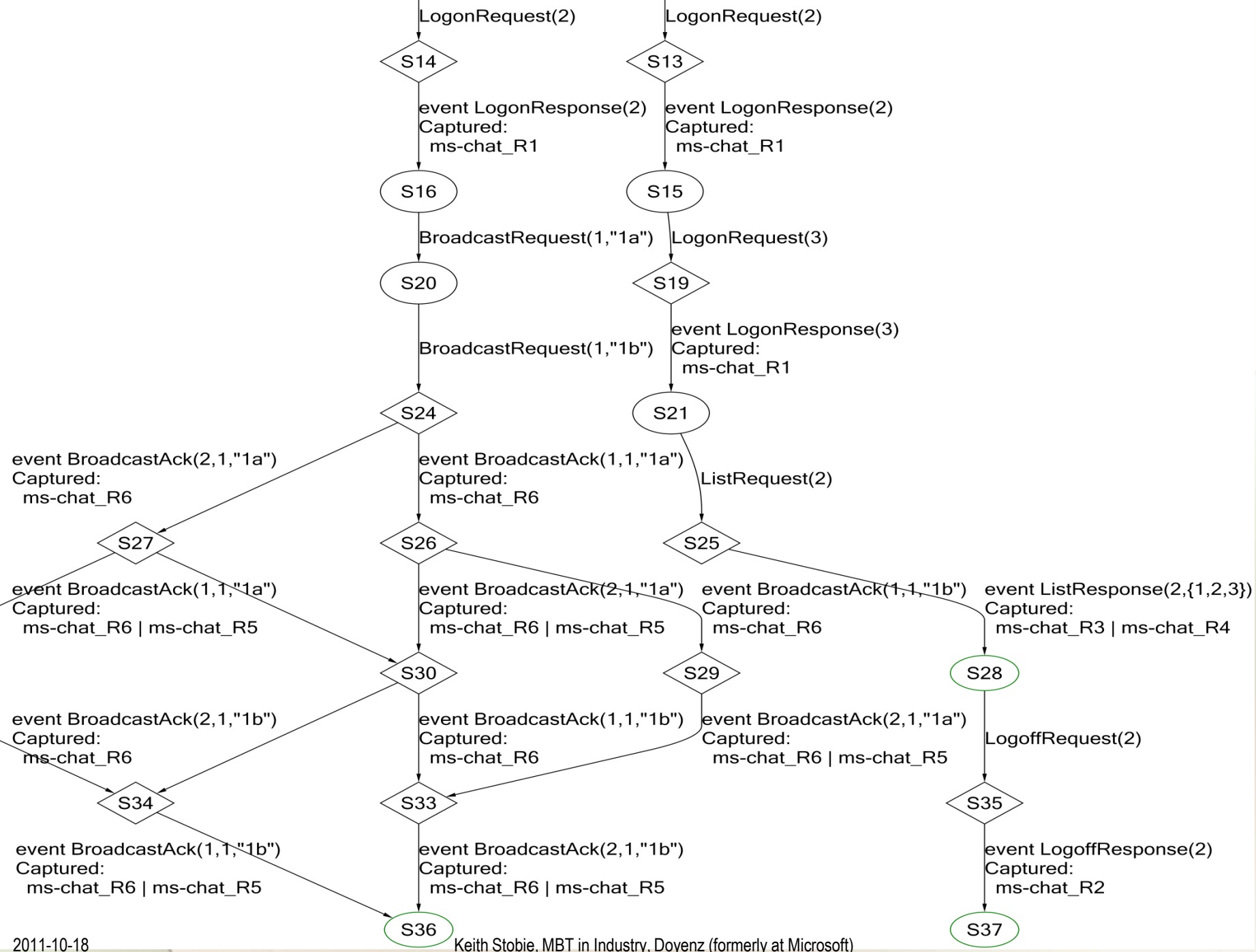
- **Requirement** : Section 3.1.5.1 - The operation MUST be failed with STATUS_OBJECT_NAME_INVALID under any of the following conditions:
  - If **PathName** is not valid as specified in [MS-FSCC] section 2.1.5.
  - If **PathName** contains a trailing backslash and **CreateOptions.FILE_NON_DIRECTORY_FILE** is TRUE.
- **Model Code (Config.cord)**

```
machine Scenario01_16() : Actions_Coverage
{
    (
        FsaInitial;
        GetSystemConfig;
        CreateFile(
                FileAttribute.READONLY,
                CreateOptions.NON_DIRECTORY_FILE,
                StreamTypeNameToOPen.INDEX_ALLOCATION,
                FileAccess.FILE_READ_DATA,
                ShareAccess.FILE_SHARE_READ,
                CreateDisposition.CREATE,
                true,
                false,
                FileType.DataFile,
                FileNameStatus.NotPathNameValid);
    )
    ||
    FSAModelProgramCoverage
}
```

- **Exploration:**

Model Code with Requirements

```
if (!caller.isAdmin) {
    RequiresCapture(1087, "In response to NetrJobGetInfo request the " +
        "server MUST Return ERROR_ACCESS_DENIED if the caller does not have " +
        "administrative privileges on the server.");
    RequiresCapture(1091, "In response to NetrJobGetInfo request, the " +
        "server MUST use Windows Error Codes as specified in [MS-ERREF].");
    return TschErrorCode.ERROR_ACCESS_DENIED;
}
else {
    //This action returns success if caller has admin privilge and
    //The requested job exists in the job list
    if (atsvcJobsStore.ContainsKey(jobId))
    {
        RequiresCapture(1025, "If the server implements the ATSvc " +
        "interface, it MUST implement the NetrJobGetInfo (Opnum 3) method.");
        RequiresCapture(1785, "NetrJobGetInfo method MUST have " +
        "administrator privileges.");
        return TschErrorCode.ERROR_SUCCESS;
    }
}
```


LogonRequest(2)
LogonRequest(2)
S14
S13
event LogonResponse(2)
Captured:
ms-chat_R1
event LogonResponse(2)
Captured:
ms-chat_R1
S16
S15
BroadcastRequest(1,"1a")
LogonRequest(3)
S20
S19
BroadcastRequest(1,"1b")
event LogonResponse(3)
Captured:
ms-chat_R1
S24
S21
event BroadcastAck(2,1,"1a")
Captured:
ms-chat_R6
event BroadcastAck(1,1,"1a")
Captured:
ms-chat_R6
ListRequest(2)
S27
S26
S25
event BroadcastAck(1,1,"1a")
Captured:
ms-chat_R6 | ms-chat_R5
event BroadcastAck(2,1,"1a")
Captured:
ms-chat_R6 | ms-chat_R5
event BroadcastAck(1,1,"1b")
Captured:
ms-chat_R6
event ListResponse(2,{1,2,3})
Captured:
ms-chat_R3 | ms-chat_R4
S30
S29
S28
event BroadcastAck(2,1,"1b")
Captured:
ms-chat_R6
event BroadcastAck(1,1,"1b")
Captured:
ms-chat_R6
event BroadcastAck(2,1,"1a")
Captured:
ms-chat_R6 | ms-chat_R5
LogoffRequest(2)
S34
S33
S35
event BroadcastAck(1,1,"1b")
Captured:
ms-chat_R6 | ms-chat_R5
event BroadcastAck(2,1,"1b")
Captured:
ms-chat_R6 | ms-chat_R5
event LogoffResponse(2)
Captured:
ms-chat_R2
S36
S37

Adapter Code Assertion/Requirements

```
// Poll for task run and terminate the process
Site.Assert.IsTrue(
        adapterHelper.IsTaskRunning(sutAdapter, execAction),
        "Task started successfully by task1");
pProcess[] procList = sutAdapter.GetProcessDetails(execAction);
Site.Assert.AreEqual(1, procList.Length,
        "One instance of task is running currently on server");

Site.CaptureRequirementIfAreEqual<string>(
    pcreds[0].userId,
    procList[0].ProcessUserName,
    4257 /* Requirement number */ ,
    "[Determine the logon type to be used depending upon "+
       "logonType supplied]If TASK_LOGON_PASSWORD is "+
       "specified, the server MUST run the task with the "+
       "user's logon and password supplied."
    );
```

Internal Adoption

Windows Services

K. Chopra : Applying MBT to Windows Web Services Software Process

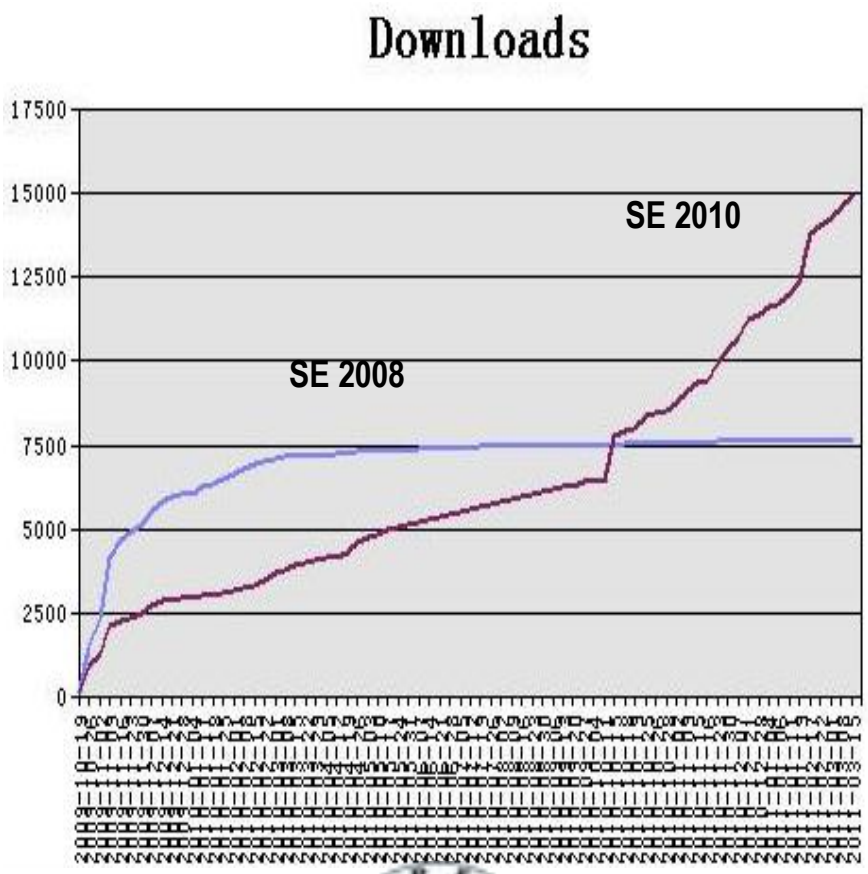
S. Mera and S. Tan, Using MBT to test Microsoft Lync Client conversation feature

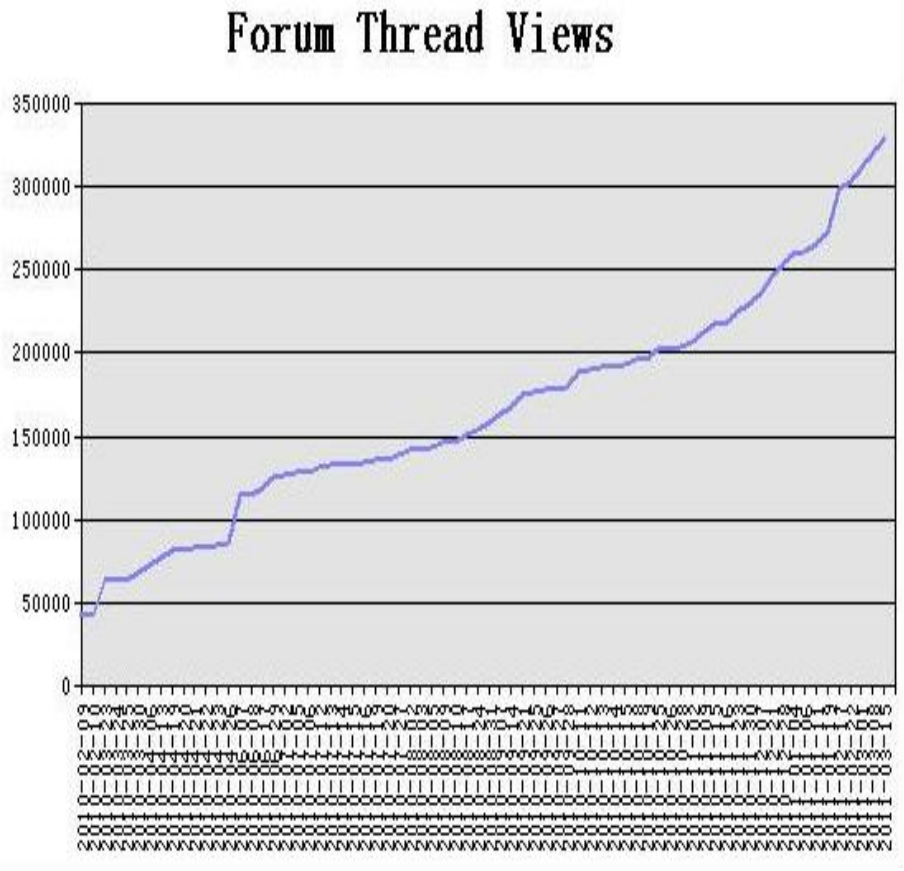
A. Mathur, Applying Hardware MBT to Cloud Scale Distributed Systems

S. Ejsing, Model Based Integration Testing

External Adoption

Microsoft Visual Studio Power Tool





Wir leben Autos.

Windows Phone

- UI Modeling
- API Modeling
  - Page navigation (Silverlight navigation framework)
  - Application Lifetime Events (pause/resume, switch)
  - Managed Camera APIs

Contact connect history

Yasser Mufti
Senior Test Lead
Windows Phone Client (WPC)

Coverage considerations

- Connect channel:
  Mobile/work/home phone, work/personal email, live/Facebook IM
- Connect type:
  Email/messages/phone call/voicemail
- Connect direction:
  Incoming/outgoing
- Connect items collapsing:
  Item type/adjacent/time group
- Connect time group:
  Today/yesterday/this week/…
- Tap target
- Linked contact (live, Facebook)
- …

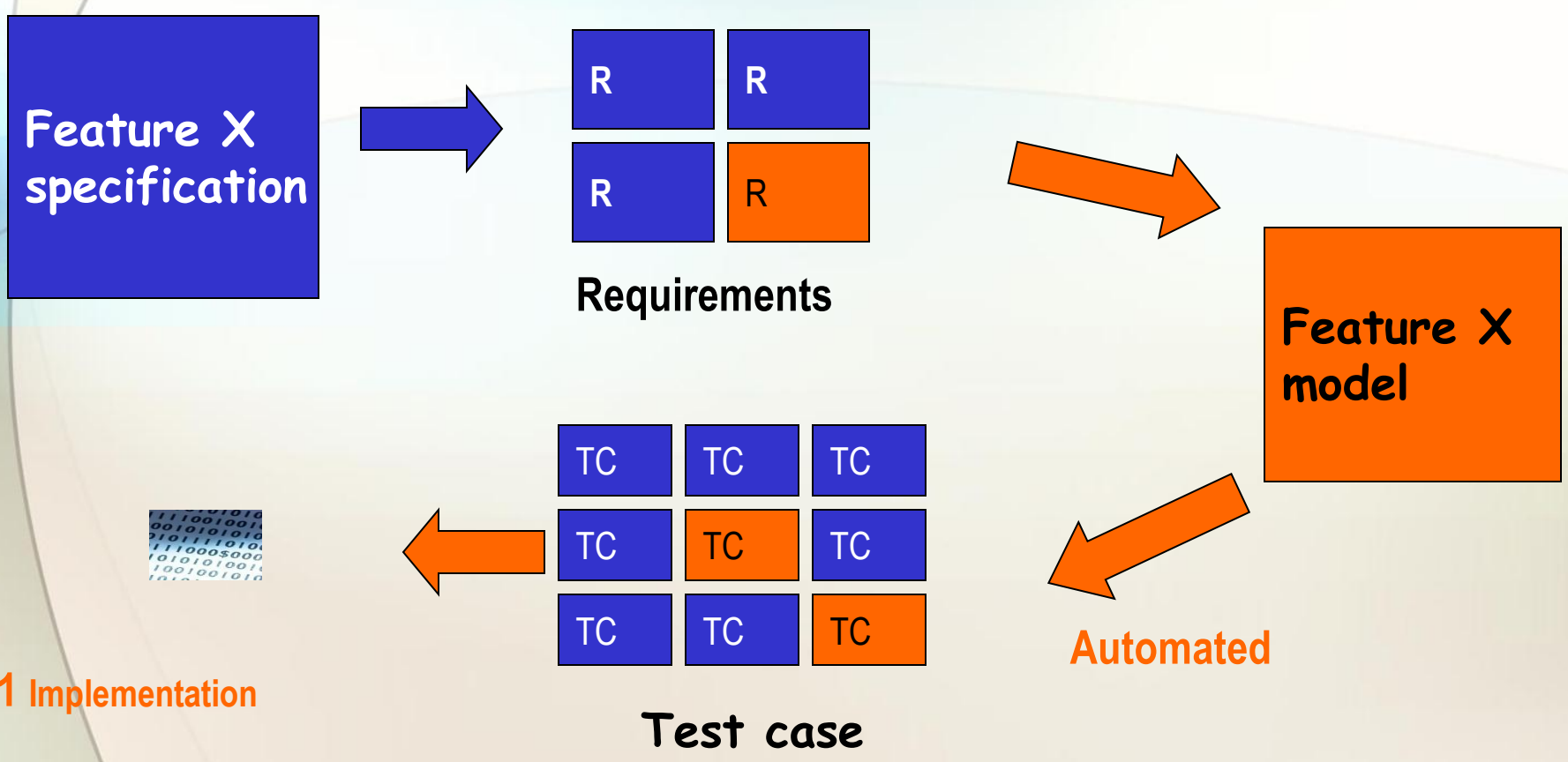
Contact connect history

Generated test case

1. See the message count go to 1
2. Go to the contact
3. Go to the history page
4. Note call & message
5. Click message
6. Message body shown
7. Go back to history
8. Click Call
9. Call made
10. End call

Yasser Mufti
Senior Test Lead
Windows Phone Client (WPC)

Maintaining a model-based test design



Test Reuse

Tests have history:

- bug finding,
- Executing time
- coverage,

```
if (!caller.isAdmin)
  return ERROR_ACCESS_DENIED;
else if (Store.ContainsKey(jobId))
  return TschErrorCode.ERROR_SUCCESS;
```

➔ Regenerating tests without unnecessarily replacing previous ones.

Office MBT Experience

- Prototype
  - **Found almost all issues** that manual testing did.
  - **Found corner issues that manual testing didn't.**
  - **SpecEx** allows to **create, design and fix models** without huge effort.
- Easier to accommodate design changes and regenerate tests
  Quality of test strategy is improved.
  Easy to inject new test scenarios (Model / Adapter more stable)
  Risky and complicated modules gain more benefit of using MBT
  Easy to prioritise and generate the tests of interest.
- Doesn't miss crucial scenarios and corner cases
- Ensures functional flow (per model) with high coverage.
- Maintenance of test scripts – Minimal

Office MBT Cons

- Right test strategy requires proper review / fine tuning of model.
- Large number of tests – Multiple failures make it hard to debug each test.(Single fix most of the time fixes all)
- Heavy learning curve.
- Highly dependent on proper working of the adapter.
- Reason some bugs not found :
    - Requirement based issue not in spec

Unleashing The Power

Spec Explorer shows its full glory when

- State is potentially infinite
- Requirements can be covered in different ways
- The system is reactive, distributed and asynchronous
- Methods have many complex parameters

Sustaining MBT

- Training & Support
- Maintenance of tools & models & adapter & tests
- Control & Visualization of the model
  - Extensive data control
- Extensibility

Brighter Future

References

- Test Model Toolkit (TMT)
  http://www.sasqag.org/pastmeetings/asml.ppt
  https://mailserver.di.unipi.it/ricerca/proceedings/ETAPS04/MBT.pdf
- Indigo uses Asml/t http://www.mi.ras.ru/~rey/modante/
  http://www.sasqag.org/pastmeetings/asml.ppt
- 2004: MS Research Spec Explorer with Spec#
  http://research.microsoft.com/en-us/projects/specexplorer/
- 2004 : Model Design Environment (MDE)
  - http://www.google.com/url?sa=t&source=web&cd=2&sqi=2&ved=0CCAQFjAB&url=http%3A%2F%2Fmodel-design-environment.software.informer.com%2F&ei=GPBeTqqMG6TViAL9h5TSDg&usg=AFQjCNFK81P-uq4ZTxbQsWPK_oO8SxTYeQ
- 2005 : Kokomo
  http://doi.ieeecomputersociety.org/10.1109/ITNG.2009.104
- 2007 : nModel http://nmodel.codeplex.com/
- Goldilocks http://www.qasig.org/presentations/QASIG_Goldilocks2.pdf

Spec Explorer 2010

Spec Explorer is available as a Visual Studio Power Tool. Its reference documentation is part of the MSDN Library. The team and community provide support through an MSDN forum. Team blogs in English and Chinese, and Channel 9 videos are used as a way to disseminate patterns and practices. There's also The full video series for a modeling class on Channel 9. The training is long (about 8 hours), but we have split it into 4 lectures (sessions). Each session has in turn 4 parts: Session 1, Session 2, Session 3, Session 4.

Protocol Test suites (some modeled):

http://www.microsoft.com/openspecifications/en/us/applied-interoperability/testing

http://channel9.msdn.com/search?term=Venkata+Maruthi+Tumuluri&type=All

Pairwise Tools

"Test Generators"

- AETG http://aetgweb.argreenhouse.com/overview.shtml
- Pairwise Independent Combinatorial Testing (PICT) http://msdn.microsoft.com/en-us/**testing**/bb980925
- Hexawise http://hexawise.com/combinatorial-and-pairwise-testing