Lessons Learned from Validating 60,000 pages of API Documentation

Requirements Engineering and Model-based Testing

Robert V. Binder
September 27, 2011
Overview

• Background
• Microsoft Protocol QA Process
  – Scope and approach
  – Requirements engineering
  – Model-based testing
• Non-Microsoft Applications
• Q & A
BACKGROUND
What is a Protocol?

• Rules for interaction between (among) endpoints using messages
  
• Data
  – Content and format

• Behavior
  – Request/Response
  – Acceptable sequences
Layers, Protocols, Stacks

- Layer: level of abstraction
- Each layer is a protocol
- Stack of layers
  - \( L \leftrightarrow L-1 \) ok
  - \( L \leftrightarrow L \pm m \) NOT ok
- Layer uses other protocols
  - HTTP over TCP or RPC
  - IP over WiFi or LAN
- Protocol may define own data or use standard format (XML)
Protocols Everywhere

- Cellular: CDMA, GSM, SMS, MMS, WAP ...
- Network: 802.11 (WiFi), 802.16 (WiMax) ...
- Wireless: Bluetooth, Zigbee, ANT, ISO 14443 ...
- Routing: OSPF, IGP, RIP, CIDR, BGP ...
- RFCs: FTP, TCP, IP, IPv4, IPv6, ... (1000s)
- WC3: SOAP, XML, HTTP, WSDL
- Corba: GIOP, IIOP, ESIOP, RMI, IDL
- FIX (Financial Information eXchange)
- Amazon API, BING API, Netflix API, Google Protocol Buffers, ...
Protocols Everywhere
SCOPE AND PROCESS
Publication by Decree

• US Federal Court and EU order
  – Microsoft to publish server side API documentation
  – Goal: interoperability for third parties

• Microsoft Open Specification Initiative
Microsoft Protocols

• Remote API for a service

• All product groups
  – Windows Server
  – Office
  – Exchange
  – SQL Server
  – Others

• 500+ protocols
  – Remote Desktop
  – Active Directory
  – File System
  – Security
  – Many others
Microsoft Technical Document (TD)

• Publish protocols as “Technical Documents”
• One TD for each protocol
• Black-box spec – no internals
• All data and behavior specified with text
Published Technical Docs


Windows Communication Protocols (MCPP)

The Microsoft Communications Protocol Program (MCPP) technical documentation set provides detailed technical specifications for protocols (including extensions to industry-standard or other published protocols) that are used by Windows Server operating systems (including Windows NT 3.1 and successors up to and including Windows Server 2008 R2) to interoperate with Windows client operating systems (including Windows 2000 Professional and successors up to and including Windows 7). The documentation set includes a set of companion overview and reference documents that supplement the technical specifications with conceptual background, overviews of inter-protocol relationships and interactions, and technical reference information, such as common data types and error codes.

Audience

The MCPP technical documentation set is intended for use in conjunction with publicly available standard specifications, network programming art, and Windows distributed systems concepts. It assumes that the reader either is familiar with this material or has immediate access to it.

The technical documentation set provides the following levels of audience support:

- For implementers: Conceptual and reference information for an implementation of one or more protocol specifications for a given task or scenario.
- For reviewers: A resource for readers who want to evaluate or understand one or more protocols.

More Information

For more information on these protocols, please visit http://www.microsoft.com/openspecifications.
Challenges

• Validation of *documentation*, not as-built implementation

• Is each TD well-formed?
  – Follows TD standards
  – Consistency, correctness, completeness

• Is each TD *all* a third party needs to develop:
  – A client that interoperates with an existing service?
  – A service that interoperates with existing clients?

• Only use over-the-wire messages
Model assertions generate and check response of actual Windows Services

Test-Driven Document Quality Assurance

• Approximates third party implementation
• Validates consistency with actual Windows implementation

 requirements specification

Analysis
Data and behavior statements

Model-based Test Suite

Lessons Learned from Validating 60,000 pages of API Documentation
© Robert V. Binder, 2011

Stobie et al, © 2010 Microsoft. Adapted with permission.
Protocol Quality Assurance Process

Authors

Test Suite

Developers

Reviewers

Plan

• Complete Test Rqmts
• High Level Test Plan

Design

• Complete Model
• Complete Adapters

Final

• Gen & Run Test Suite
• Prep User Doc

Study

• Scrutinize TD
• Define Test Strategy

Review

• TD ready?
• Strategy OK?

Review

• Test Rqmts OK?
• Plan OK?

Review

• Model Ok?
• Adapter Ok?

Review

• Coverage OK?
• Test Code OK?

Lessons Learned from Validating 60,000 pages of API Documentation
© Robert V. Binder, 2011
Cast of Thousands

- Hard milestones/deadlines imposed by Federal Judge
- “Technical Committee” chartered by court, with over 50 full time reviewers of published TDs
- 100s of senior MSFT developers wrote/revised TDs
- TD publication staff
- More than 350 test suite devs (mostly in Hyderabad & Beijing)
- ~20 Independent Reviewers (5 System Verification Assoc.)
- Process Architects (MSFT & System Verification Assoc.)
- MSFT Netmon and other tool developers
- MSFT Plugfest team
- MSFT project management
Results

• Published 500+ TDs
  – 60,000+ pages
• 50,000+ “Technical Document Issues”
  – Most identified before tests run
• Many Plugfests, many 3rd party users
• Released high interest test suites as open source
• Met all regulator requirements, on time
  – Judge closes DOJ anti-trust case May 12, 2011
REQUIREMENTS ENGINEERING
TD Statements

Data Statement

2.2.4 DHCPv6 Option Code 15 (0x000F) - User Class Option

DHCPv6 clients implementing this specification MUST use the message format and semantics specified in [RFC3315] when sending a User Class Option to a DHCPv6 server.

Behavior Statement

3.2.7.1 DhcpAppendVendorSpecificOption

When the cumulative size of all the vendor-specific options being sent in a message exceeds 255 bytes, the Microsoft Encoding Long Options Packet, specified in section 2.2.9, MUST be used.

Endnote

<24> Windows XP and Windows Server 2003 DHCP clients request only option code 249 in the Parameter Request Request List.
<table>
<thead>
<tr>
<th>Req ID</th>
<th>Doc Sect</th>
<th>Description</th>
<th>Pos</th>
<th>Neg</th>
<th>Deriv</th>
<th>Inform/Norm</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSCH</td>
<td>R113</td>
<td>The column field of the TASK_XML_ERROR_INFO structure MUST contain the column where parsing failed.</td>
<td>R11</td>
<td>R31</td>
<td>Norm</td>
<td>Adapter</td>
<td></td>
</tr>
<tr>
<td>TSCH</td>
<td>R142</td>
<td>The client MUST set the File Version (2bytes, it contains the Version of the .JOB file format) field of the FIXDLEN_DATA structure to 0x0001.</td>
<td>R11</td>
<td>R31</td>
<td>Norm</td>
<td>Non-testable</td>
<td></td>
</tr>
<tr>
<td>TSCH</td>
<td>R145</td>
<td>The server MUST ignore the value in the App Name Len Offset field of the FIXDLEN_DATA structure.</td>
<td>R11</td>
<td>R31</td>
<td>Norm</td>
<td>Test Case</td>
<td></td>
</tr>
<tr>
<td>TSCH</td>
<td>R146</td>
<td>The Trigger Offset (2 bytes) field of the FIXDLEN_DATA structure MUST contain the offset in bytes within the .JOB file where the task triggers are located.</td>
<td>R11</td>
<td>R31</td>
<td>Norm</td>
<td>Adapter</td>
<td></td>
</tr>
<tr>
<td>TSCH</td>
<td>R1332</td>
<td>Upon receipt of the SchRpcGetSecurity call, the server MUST:</td>
<td>R11</td>
<td>R64</td>
<td>Norm</td>
<td>Unverified</td>
<td></td>
</tr>
<tr>
<td>TSCH</td>
<td>R1333</td>
<td>[Upon receipt of the SchRpcGetSecurity call, the server MUST] Return the value 0x8007007B, the HRESULT form of the Win32 error ERROR_INVALID_NAME, if the specified path is not in the format specified in section 2.3.11.</td>
<td>R11</td>
<td>R64</td>
<td>Norm</td>
<td>Test Case</td>
<td></td>
</tr>
<tr>
<td>TSCH</td>
<td>R1335</td>
<td>Through the SchRpcEnumFolders method if client requests items 1 - 10 and then 11 - 20, the second call MAY return duplicate entries if the folder list has changed in between calls.</td>
<td>R11</td>
<td>R50</td>
<td>Inform</td>
<td>Unverified</td>
<td></td>
</tr>
<tr>
<td>TSCH</td>
<td>R1337</td>
<td>The SchRpcEnumFolders method MUST retrieve a list of folders on the server.</td>
<td>R11</td>
<td>R50</td>
<td>Inform</td>
<td>Unverified</td>
<td></td>
</tr>
<tr>
<td>TSCH</td>
<td>R1350</td>
<td>Upon receipt of the SchRpcEnumFolders call, the server MUST:</td>
<td>R11</td>
<td>R50</td>
<td>Inform</td>
<td>Unverified</td>
<td></td>
</tr>
</tbody>
</table>
Document Debugging

• Every TD statement analyzed
  – Scrutinize
  – Categorize
  – Make context explicit
  – Trace dependencies
  – Assess testability
  – Allocate
Scrubinize

- Ambiguous phrasing
- Misuse of MUST, SHOULD, MAY
- Inconsistent
- Unclear
- TD template violations
- *Write bug report for author correction*
Categorize

• Normative or Informative?
  – Like code comments (informative)?
    Conceptually, cells are numbered in a dataset as if the dataset were a $p$-dimensional array, where $p$ is the number of axes.
  – Or like code (normative)?
    **SVR_RESP (1 byte):** A single byte whose value MUST be 0x05.
    – If removed, would that prevent 3rd party interop?
• No modeling/testing for informative
Make Context Explicit

- Add implicit antecedents
- Use \[\] to indicate addition
- Preserves meaning in code, test results, log files

<table>
<thead>
<tr>
<th>Original TD statements</th>
<th>If the computeByClause is present, one group is created for each unique combination of values in the column or columns specified in the computeByClause. Otherwise, all rows of the child RecordSet are treated as a single group.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Requirement 1</td>
<td>If the computeByClause is present, one group is created for each unique combination of values in the column or columns specified in the computeByClause.</td>
</tr>
<tr>
<td>Test Requirement 2</td>
<td>Otherwise [[if the computeByClause is not present]], all rows of the child RecordSet are treated as a single group [in the computeByClause.]</td>
</tr>
</tbody>
</table>
Trace Dependencies

• Is there a stated observable effect:
  – For every cause?
  – When a cause is missing or corrupted?

• Record analysis with **linked requirements**

<table>
<thead>
<tr>
<th>Req ID</th>
<th>Description</th>
<th>Pos</th>
<th>Neg</th>
<th>Derived</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>R100</td>
<td>Actions: This part MUST be present and MUST specify the action to be performed once the task is started.</td>
<td>R110</td>
<td>???</td>
<td></td>
<td>Test Case</td>
</tr>
<tr>
<td>R110</td>
<td>The server MUST execute multiple actions sequentially, in the order specified in the Actions field.</td>
<td></td>
<td></td>
<td></td>
<td>Test Case</td>
</tr>
</tbody>
</table>
Assess Testability

• A test requirement is testable if:
  – Sufficient to generate and/or evaluate in code
  – Observable over-the-wire

• Non-testable if:
  – Unobservable
  – Uncontrollable
  – Infeasible
  – Excessive cost to develop test
Assess Testability

• Unobservable or uncontrollable
  All the structures MUST begin on 8-byte boundaries, although the data that is contained within the structure need not be aligned to 8-byte boundaries.
  – *Can’t detect mis-alignment from test endpoint*
  After close, the server MUST release all resources.
  – *No way to check using protocol*

• Infeasible
  The server MUST return a unique ID.
  – *No way to conclusively determine uniqueness*
Assess Testability

• What to do about non-testable statements?
  – Punt?
    • Interpretation unpredictable (testers and users)
  – Skip?
    • Taints credibility, lowers coverage
• Add *derived test requirement*
  – *Rewrite* non-testable
  – Strictly limited revision or elaboration
  – Original statement preserved, not tested
• *Significant requirements engineering innovation*
# Derived Test Requirements

<table>
<thead>
<tr>
<th>Req ID</th>
<th>Description</th>
<th>Derived</th>
<th>Verification</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>R42</td>
<td>MUST accept <em>any</em> positive number</td>
<td></td>
<td>Non-testable</td>
<td>Infeasible</td>
</tr>
<tr>
<td>R1042</td>
<td>MUST accept 1024</td>
<td>42:c</td>
<td>Test Case</td>
<td></td>
</tr>
<tr>
<td>R39</td>
<td><strong>Ignored</strong> by the server on receipt</td>
<td></td>
<td>Non-testable</td>
<td>Server internal behavior</td>
</tr>
<tr>
<td>R1039</td>
<td>Reply is the same whether 0 or non-zero is used for Field</td>
<td>39:p</td>
<td>Test Case</td>
<td></td>
</tr>
</tbody>
</table>

- **Case**: A selected instance of a domain
- **Partial**: Observable effect
- **Inferred**: Result of several causes
# Fully Elaborated Test Requirements

## Requirements Table

<table>
<thead>
<tr>
<th>Req ID</th>
<th>Description</th>
<th>Pos</th>
<th>Neg</th>
<th>Derived</th>
<th>Verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>R100</td>
<td><strong>Actions</strong>: This part MUST be present and MUST specify the action to be performed once the task is started.</td>
<td>R110</td>
<td>R1100</td>
<td>Test Case</td>
<td></td>
</tr>
<tr>
<td>R110</td>
<td>The server MUST execute multiple actions sequentially, in the order specified in the Actions field.</td>
<td></td>
<td></td>
<td>Test Case</td>
<td></td>
</tr>
<tr>
<td>R113</td>
<td><strong>pErrorInfo</strong>: If this parameter is non-NULL and the XML task definition is invalid, the server MUST return additional error information.</td>
<td></td>
<td></td>
<td>Test Case</td>
<td></td>
</tr>
<tr>
<td>R114</td>
<td>0x8004131A <strong>SCHED_E_MISSINGNODE</strong>: The task XML is missing a required element or attribute.</td>
<td></td>
<td></td>
<td>Test Case</td>
<td></td>
</tr>
<tr>
<td>R1100</td>
<td>If <em>Action</em> is missing, <strong>SCHED_E_MISSINGNODE</strong> is returned in <strong>pErrorInfo</strong></td>
<td></td>
<td></td>
<td>Test Case</td>
<td></td>
</tr>
</tbody>
</table>

Stobie et al, © 2010 Microsoft. Adapted with permission.
Allocate Test Requirements

• To Test Case
  – Develop model contract and/or test code
    • Generate the condition and send a message
    • Evaluate response (pass or fail)

• To Adapter
  – Data structure/format checked as side-effect
    • Netmon parsing
    • Transport layer marshaling
## Document Debugging

<table>
<thead>
<tr>
<th>Bug</th>
<th>Fix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template, MUST-SHOULD-MAY wrong</td>
<td>TDI – Author rewrite</td>
</tr>
<tr>
<td>Ambiguous, unclear, inconsistent</td>
<td>TDI – Author rewrite</td>
</tr>
<tr>
<td>Missing or incorrect</td>
<td>TDI – Author rewrite</td>
</tr>
<tr>
<td>SUT response inconsistent</td>
<td>TDI – Production bug and/or author rewrite</td>
</tr>
<tr>
<td>Implicit antecedent</td>
<td>Add antecedent</td>
</tr>
<tr>
<td>Cause or effect too broad</td>
<td>Add derived requirement, narrow domain</td>
</tr>
<tr>
<td>No effect for corrupt/missing cause</td>
<td>Add derived “negative” requirement</td>
</tr>
<tr>
<td>Unobservable or uncontrollable</td>
<td>Add derived with observable effect</td>
</tr>
<tr>
<td>Infeasible</td>
<td>Add derived with narrowed scope</td>
</tr>
</tbody>
</table>

Lessons Learned from Validating 60,000 pages of API Documentation

© Robert V. Binder, 2011
MODEL-BASED TESTING
Model-Based Testing in a Nutshell

Requirements

Develop

Test

Feedback

Generate

Test Suite

Inputs (Test Sequences)

Expected Outputs (Test Oracle)

Control

Observe

Implementation Under Test

Produce

Verdict

Stobie et al, © 2010 Microsoft. Adapted with permission.
Why Model-based Testing?

• Scope
  – Automate generation of huge number of tests
  – Mitigate brittleness/breakage risk

• Highly structured behavior well-suited to modeling

• Easier to assess model than huge test suite

• Consistent and automatic transition coverage versus arbitrary or ad hoc strategies
Spec Explorer

• Model-based testing tool
  – Developed at Microsoft Research
  – Productized after extensive use
  – Fully integrated with Visual Studio Test

• Development UI

• Generates standalone executable test suite
Spec Explorer Personality

• Entire model in C# - no pictures/UML
  – Inline Spec Explorer pragmas/calls
  – Include any programmable function or Dot Net capability

• Aggregate model synthesized
  – State machine slicing defines scenarios

• Coverage strategy
  – All transitions of the explored model/scenario, short or long path
  – Combinational selection of transition parameter values
static class Model {
    public enum TimerMode { Reset, Running, Stopped }
    static bool displayTimer = false;
    static TimerMode timerMode = TimerMode.Reset;
    static bool timerFrozen = false;

    [Action] static void StartStopButton() {
        Contracts.Requires(displayTimer);
        if (timerMode == TimerMode.Running) {
            timerMode = TimerMode.Stopped;
            timerFrozen = false;
        } else {
            timerMode = TimerMode.Running;
        }
    }
    [Action] static void ModeButton() { ... }
    [Action] static void ResetLapButton() { ... }
    [AcceptingState] static bool IsTimerReset() { ... }
    ...
}
Model Exploration

• Modeler defines *scenarios*
  – Subset of all *[Action]* methods
  – Represent use cases or other slice
  – Manages state explosion problem

• SE explores model
  – Constraint solver finds feasible paths using initial data values and symbolic execution
  – Supports iterative model development

• SE generates send/evaluate messages (test suite)
  – Mix generated and hand-coded tests
  – Standalone or on-the-fly
Typical Test Configuration

Lessons Learned from Validating 60,000 pages of API Documentation
© Robert V. Binder, 2011
if (!caller.isAdmin) {
    RequiresCapture(1087, "In response to NetrJobGetInfo request, the " + "server MUST Return ERROR_ACCESS_DENIED if " + "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 privilege 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;
    }
}
Netmon Capture with Reqmts

20540TSAP TSAP:TestCase Name=....Test_ITask_RegisterFlagsS8, Message= MS-TSCH_R1215
20541TSAP TSAP:TestCase Name=....Test_ITask_RegisterFlagsS8, Message= MS-TSCH_R1569
20544TSCH TSCH:ITaskSchedulerService::SchRpcDelete Response, ReturnValue=1 vstesthost.exe
20545TSCH TSCH:ITaskSchedulerService::SchRpcGetTaskInfo Request, Path=CH\122330325 Flags=0 (0x0)
20546TSCH TSCH:ITaskSchedulerService::SchRpcGetTaskInfo Response, Enabled=0 State=0 ReturnValue=1
20547TSAP TSAP:TestCase Name=....Test_ITask_RegisterFlagsS8, Message= MS-TSCH_R5
20548TSAP TSAP:TestCase Name=....Test_ITask_RegisterFlagsS8, Message= MS-TSCH_R17
20549TSAP TSAP:TestCase Name=....Test_ITask_RegisterFlagsS8, Message= MS-TSCH_R10
20550TSAP TSAP:TestCase Name=....Test_ITask_RegisterFlagsS8,
   Message=Assert.IsTrue succeeded. The SchRpcDelete method MUST delete a task in the task store.
20551TSAP TSAP:TestCase Name=....Test_ITask_RegisterFlagsS8, Message= MS-TSCH_R1892
20552TSAP TSAP:TestCase Name=....Test_ITask_RegisterFlagsS8,
   Message=Assert.IsTrue succeeded. Upon receipt of the SchRpcDelete call the server MUST delete the task
   from the XML task store.
20553TSAP TSAP:TestCase Name=....Test_ITask_RegisterFlagsS8, Message= MS-TSCH_R1496
20554TSAP TSAP:TestCase Name=....Test_ITask_RegisterFlagsS8, Message= MS-TSCH_R1211
20555TSAP TSAP:TestCase Name=....Test_ITask_RegisterFlagsS8, Message= MS-TSCH_R1497
20557TSAP TSAP:TestCase Name=....Test_ITask_RegisterFlagsS8, Status=Passed

Stobie et al, © 2010 Microsoft. Adapted with permission.
Complete Traceability

- Technical Document
- Requirements Spec
- Model
- Test Suite
- Logs
- Network Captures

Stobie et al, © 2010 Microsoft. Used with permission.
On average, model-based testing took 42% less time than hand-coding tests.

<table>
<thead>
<tr>
<th>Task</th>
<th>Avg Hours Per Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>TD review</td>
<td>1.1</td>
</tr>
<tr>
<td>Requirement gathering</td>
<td>0.8</td>
</tr>
<tr>
<td>Model authoring</td>
<td>0.5</td>
</tr>
<tr>
<td>Traditional test coding</td>
<td>0.6</td>
</tr>
<tr>
<td>Adapter development</td>
<td>1.2</td>
</tr>
<tr>
<td>Test case execution</td>
<td>0.6</td>
</tr>
<tr>
<td>Final adjustments</td>
<td>0.3</td>
</tr>
<tr>
<td>Total, all phases</td>
<td>5.1</td>
</tr>
</tbody>
</table>

Grieskamp et al.
LESSONS FOR APPLICATION TO NON-MICROSOFT PROTOCOLS
Interoperability Matters

Standards View

Client Endpoint
- WSDL
- SOAP
- HTTP
- TCP
- IP
- 802.11

Server Endpoint
- WSDL
- SOAP
- HTTP
- TCP
- IP
- 802.11

Real World

Lessons Learned from Validating 60,000 pages of API Documentation
© Robert V. Binder, 2011
Interoperability Matters

... as suppliers tried to expand applications of Bluetooth, it proved hard to ensure that all enabled devices really worked with each other.

By contrast, IEEE 802.11b wireless LANs had a smoother road because of rigorous certification by the Wireless Ethernet Compatibility Alliance.

• Ubiquitous problem
• How can RE/MBT help?
Increase Protocol Traction

• Publisher: Use RE/MBT for doc, reference impl, conformance test suite

• Users: High quality doc + complete test suite = faster, better, cheaper
Facilitate User Extension

- RE/MBT is extensible
- User extends conformance test suite
  - *Reduce time and cost*
  - *Improve interoperability*
  - *Drives community traction*
Facilitate Acceptance Testing

- User develops impl for customer
- Customer conducts acceptance using RE/MBT conformance test suite
  - Objective criteria
  - Avoids gap disputes
  - Reduces time and cost for all

Lessons Learned from Validating 60,000 pages of API Documentation
© Robert V. Binder, 2011
System Verification Associates

• Protocol V&V
  – Validate Documentation
  – Verify Implementation
  – Conformance Testing
  – Interoperability Testing
  – Process Architect

Bob Binder
rvbinder@gmail.com
312 404 5341
www.sysvas.com
Twitter: @rvbinder
Resources and Sources

• Microsoft Open Specification web site
  – http://www.microsoft.com/openspecifications
• Project Overview
  – http://queue.acm.org/detail.cfm?id=1996412
• Spec Explorer
• Netmon and protocol parsers
• Protocol Test Suites (must provide Live Id to login)
  – https://connect.microsoft.com/site1165/Downloads
