Automated Test Design
with TTCN-3

TTCN-3 User Conference
Beijing, June 8th 2010

Conformiq
Tutorial
Conformiq, Inc

• Founded in 1998
• Privately held
• World locations:
  – Saratoga, CA, USA (HQ)
  – Helsinki, Finland (R&D)
  – Stockholm, Sweden (Nordic)
  – Munich, Germany (EU)
  – Hyderabad, India through our partner
Tutorial Outline

• Automated Test Design
• Why Automated Test Design?
• Conformiq Designer
• Conformiq Designer and TTCN-3
• SIP Example Walkthrough
Automated Test Design
Challenges of Manual Test Design

- Missed tests
  - Can result in product defects

- Incorrect tests
  - Cause additional test development work

- Redundant tests
  - Cause extra development and maintenance costs

- Unknown requirements coverage
  - Can result in untested features

- Frequent changes to specification
  - Cause high cost for test suite maintenance
Evolution of Test Design

Manual Design Manual & Automatic Execution

- Experiment and test manually
- Collect test cases
- Design tests from specification
- Write a model of specification
- Write test scripts
- Design tests and derive test scripts automatically
- Run scripts for initial testing
- Run scripts for regression testing
- Write test scripts
- Write report
# What is Model-Based Testing?

<table>
<thead>
<tr>
<th>Approach</th>
<th>System model driven</th>
<th>Graphical test case design</th>
<th>Test model driven</th>
</tr>
</thead>
<tbody>
<tr>
<td>What the user models</td>
<td>Expected behavior of the SUT</td>
<td>Individual test cases</td>
<td>Structure and expected behavior of the environment that the SUT is embedded in</td>
</tr>
<tr>
<td>How data sent by the test system is determined</td>
<td>Automatically</td>
<td>Have to be manually defined by the user</td>
<td>Defined via testing strategy</td>
</tr>
<tr>
<td>How data sent by the SUT is validated</td>
<td>Expected test data and verdicts are derived automatically</td>
<td>Expected test data and verdicts have to be defined manually</td>
<td>Expected data and verdicts are defined via testing strategy and modeling</td>
</tr>
<tr>
<td>How test cases are traced to requirements</td>
<td>Can be done automatically if the user includes requirement annotations in the model</td>
<td>Tracing has to be specified as part of every test case</td>
<td>Can be done automatically if the user includes requirement annotations in the model</td>
</tr>
<tr>
<td>How tests are maintained</td>
<td>Changes to the model are automatically propagated to all tests</td>
<td>Each test has to be individually and manually maintained</td>
<td>Testing strategies and oracles need to be maintained by hand</td>
</tr>
<tr>
<td>Can it produce TTCN-3 code</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Can end-to-end tests be easily derived from conformance testing artifacts</td>
<td>Yes, straightforwardly</td>
<td>Usually no, because test logic and data needs to be changed</td>
<td>Usually no, because test models can not be easily composed</td>
</tr>
<tr>
<td>What model complexity of is</td>
<td>High</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>What tasks are eliminated</td>
<td>Design test cases</td>
<td>Write executable tests</td>
<td>Write executable tests</td>
</tr>
<tr>
<td></td>
<td>Maintain test cases</td>
<td>Maintain test case traceability</td>
<td></td>
</tr>
</tbody>
</table>
Automated Test Design

• Model Based Testing (MBT)
  – An "umbrella" of approaches that can be used to generate tests from models

• Automated Test Design (ATD)
  – An approach that uses system model driven MBT to design, document, and implement tests

• Enables
  – Faster test development
  – Improved test quality
  – Wider test coverage & guaranteed requirement coverage
  – Cost-effective test maintenance
  – Earlier test validation & detection of specification defects
  – Independence from test execution environment
Integration of Automated Test Design

Test execution engine and adaptation can be reused “as is”!
Why Automated Test Design?
Manual vs. Automated Test Design

Waterfall
Manual
- Design tests
- Write test cases
- Run tests

Waterfall
Automated
- Write model
- Run tests

Agile
Automated

→ time

Find and Fix more defects sooner
Productivity Improvement

- **1X** Manual test design
- **5X** Automated Test Design in initial deployment
- **10–20X** Automated Test Design in subsequent tested product iterations

Source: average results from customer benchmarks
10–20X Elaborated

- Higher test quality
- The same model is the source of all tests → easier to maintain
- Models are easier to review and communicate than test scripts
- Model components are easier to reuse, share and compose than test cases which are ”snapshots”
Productivity Improvement as an Enabler

+400% productivity increase in test design (average of customer benchmarks)

- Less defects found by customer
- More available resources
- Shorter turnaround time
- Faster integration process
Basic Benchmarking Method

**Begin**
- Start from the same documentation that was used previously for creating hand-written test cases

**Model**
- Create a model and generate tests

**Judge**
- Customer expert decides when an equal test coverage has been reached

**Measure**
- Compare man-hours spent on test design
Conformiq Designer
ATD with Conformiq Designer

• Reads in system models and coverage criteria
• Automatically designs test input and expected output data and timer handling
• Renders automatically generated tests in chosen output format
• Imports models from 3rd party tools
• Integrated into Eclipse
The System Model…

• Describes the correct (expected) operation of the IUT
• Should be kept as abstract as testing objectives
• Specified using Conformiq Modeling Language (QML)
• Is processed as an object-oriented computer program
Conformiq Modeling Language (QML)

- System Block
- UML Diagrams
- Java-like Action Language

QML Model
From the System Model to the "Black Box"
# Coverage Criteria supported by CQ Designer

<table>
<thead>
<tr>
<th>Name</th>
<th>Explanation</th>
<th>Typically Used For</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements Coverage</td>
<td>Cover every “requirement” statement</td>
<td>Requirements traceability</td>
</tr>
<tr>
<td>State Coverage</td>
<td>Cover every state in every state chart</td>
<td>Basic test generation</td>
</tr>
<tr>
<td>Transition Coverage</td>
<td>Cover every transition (from one state to another) in every state chart</td>
<td>Basic test generation</td>
</tr>
<tr>
<td>Condition Coverage</td>
<td>Cover both “true” and “false” case of if’s and similar conditional constructs</td>
<td>Basic test generation</td>
</tr>
<tr>
<td>Parallel Transition Coverage</td>
<td>Cover every interleaving of two independent transitions in multi-threaded models</td>
<td>Feature interaction</td>
</tr>
<tr>
<td>Switch Coverage</td>
<td>Cover every combination of the entry and exit transitions of all states</td>
<td>Extended test generation</td>
</tr>
<tr>
<td>Atomic Condition Coverage</td>
<td>For Boolean connectives, cover all combinations of left and right truth values (taking short-circuit evaluation into account)</td>
<td>Extended test generation</td>
</tr>
<tr>
<td>Boundary Value Analysis</td>
<td>For comparisons of integer values, cover boundary conditions</td>
<td>Extended test generation</td>
</tr>
<tr>
<td>Method Coverage</td>
<td>Cover every method declared</td>
<td>Extra structural traceability</td>
</tr>
<tr>
<td>Statement Coverage</td>
<td>Cover every statement</td>
<td>Extra structural traceability</td>
</tr>
<tr>
<td>Transition All Paths</td>
<td>Cover all arbitrarily long distinct paths through transitions—requires a terminating model</td>
<td>Exhaustive test generation</td>
</tr>
<tr>
<td>Control Flow All Paths</td>
<td>Cover all arbitrarily long control flow paths—requires a terminating model</td>
<td>Exhaustive test generation</td>
</tr>
</tbody>
</table>
## Conformiq Designer Features

<table>
<thead>
<tr>
<th>Mathematically Generates</th>
<th>Other Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Test inputs</td>
<td>• Modeling in UML and Java-like notation</td>
</tr>
<tr>
<td>• Expected test outputs</td>
<td>• Multiple, fully customizable output formats</td>
</tr>
<tr>
<td>• Test timings</td>
<td>• Import of UML diagrams from 3rd party tools</td>
</tr>
<tr>
<td>• Sequence charts</td>
<td>• Interactive workbench</td>
</tr>
<tr>
<td>• Executable test cases</td>
<td>• Integrated in Eclipse® framework</td>
</tr>
<tr>
<td>• Traceability matrices</td>
<td></td>
</tr>
<tr>
<td>• Test dependency matrices</td>
<td></td>
</tr>
</tbody>
</table>

Copyright © Conformiq Inc. and its subsidiaries. All rights reserved.
Conformiq Designer Applicability

Regression Testing

Functional Testing

Component Testing

Integration Testing

System Testing

Usability Testing

Robustness Testing

Performance Testing
Conformiq Designer and TTCN-3
Conformiq and TTCN-3

• Conformiq Designer ships with an out-of-the-box TTCN-3 generator

• Starting with Conformiq Designer 4.2 support for import of TTCN-3 types and constants for model specification

• Company has provided support for TTCN-3 generation since 2002

• Active in ETSI’s Technical Committee Methods for Testing and Specification (TC MTS)
  – Home of TTCN-3
Experiences with TTCN-3 Tools

- **MessageMagic** (Elvier)
- **Titan** (Ericsson proprietary)
  - See T. Funke’s presentation at SQC 2009
- **General Test Runner (GTR)** (Huawei proprietary)
  - See X. Gao’s paper at TESTCOM 2008
- **TTworkbench** (TestingTech)
- **Tau Tester** (Telelogic)
  - Now part of IBM’s offering
Why Combine CQ Designer with TTCN-3?

• Benefits for TTCN-3 users:
  ✅ Automated test case design, writing, and documentation
  ✅ Consistent test design and quality
  ✅ Guaranteed requirement coverage
  ✅ More efficient test suite maintenance
  ✅ Easier test artefact review, reuse and sharing
  ✅ Reuse existing test execution platforms

• Benefits for Conformiq Designer users:
  ✅ Well-defined internationally standardized testing language and interfaces to execution platforms
  ✅ Application and test tool independent
  ✅ Well known and accepted in industry
  ✅ Automatic test execution
Test Interfaces: From model to real system
A closer look at the TTCN-3 Test Cases
A Test Harness Implementation Example

- Conformiq Designer generates TTCN-3 function calls (in test case)

  \[
  \text{f\_send\_SIPRequest\_to\_netIn( c\_SIPRequest15 );}
  \]

- The function is implemented by the TTCN-3 Harness

  \[
  \begin{align*}
  &\text{// Finalize and transform SIP request to TTCN-3 type} \\
  &\text{// structure if different from the one used by model} \\
  &\text{function f\_send\_SIPRequest\_to\_netIn( in SIPRequest p\_req )} \\
  &\text{runs on CQ\_MTC} \\
  &\{ \\
  &\quad \text{var TTCN3\_SIP\_Request v\_TTCNReq;} \\
  &\quad v\_TTCNReq := f\_prepare\_send\_SIPRequest( p\_req ); \\
  &\quad \text{netIn.\_send( v\_TTCNReq );} \\
  &\}
  \end{align*}
  \]
function f_prepare_send_SIPRequest ( in SIPRequest p_req )
runs on CQ_MTC return TTCN3_SIP_Request
{
    // 1. Finalize headers
    p_req.callId := f_send_add_nonce_to_callId( p_req.callId );
    p_req.CSeq := f_send_add_nonce_to_cSeq( p_req.Cseq );
    p_req.from_ := f_send_add_nonce_to_from_tag( p_req.from_ );

    // 2. Replace CQ Designer generated symbolic values in headers
    // with values at runtime
    p_req.via := f_send_restore_via_branch( p_req.via );
    p_req.to_ := f_send_restore_to_tag( p_req.to_ );

    // transform from abstract to TTCN-3 type structure if needed
    return f_SIPRequest_transform2t3( p_req );
}
Test Harness - Preparation after Receiving

```java
function f_prepare_and_match_SIPRequest ( in SIPRequest p_expReq,
                                         in TTCN3_SIP_Request p_rcvTTCNReq )
runs on CQ_MTC return SIPRequest
{
    // 1. transform from TTCN-3 to abstract type structure if needed
    var SIPRequest v_rcvReq := f_SIPRequest_transform2cq( p_rcvTTCNReq );

    // 2. Store key values later needed in sending and replace
    // them with "generated values" for matching.
    v_rcvReq.via := f_recv_store_via_branch( v_expReq.via,v_rcvReq.via );
    v_rcvReq.to_ := f_recv_store_to_tag( v_expReq.to_, v_rcvReq.to_ );

    // 3. For matching purposes replace runtime header information
    v_rcvReq.callId := f_recv_remove_nonce_from_callId( v_rcvReq.callId );
    v_rcvReq.CSeq := f_recv_remove_nonce_from_cSeq( v_rcvReq.Cseq );
    v_rcvReq.from_ := f_recv_remove_nonce_from_from( v_rcvReq.from_ );
    return v_rcvReq;
}
```
Testing of a SIP User Agent Client: a Walkthrough
Testing of a SIP User Agent

• Task:
  Test basic call functionality of a SIP User Agent Client

• Basis:
  Create system model directly from IETF RFC 3261 “SIP: Session Initiation Protocol”

• System Under Test:
  A normal phone or a soft client
SIP User Agent and its Environment

Make a call to...

User

User Interfaced

SUT

SIP User Agent Client

Network

Network Interface

SIP User Agent Server

SIP Proxy
Tested Functionality

• Call establishment (“SIP INVITE”)
• Call termination (“SIP BYE”)
  – caller-initiated
  – callee-initiated
• Call cancelation (“SIP CANCEL”)
• Timers
  – re-transmission
  – transaction
Modeled Requirements

The SIP User Agent Client must:

1. Establish a session with SIP ACK request
2. Terminate a session with SIP BYE request
3. Confirm a SIP BYE request with a SIP 200 OK response
4. Re-send an SIP INVITE request after timeout A
5. Terminate an SIP INVITE request after timeout B
6. Re-send a SIP BYE request after timeout E
7. Re-send SIP CANCEL request after timeout E
8. Terminate a SIP BYE request after timeout F
9. Terminate a SIP CANCEL request after timeout F
The Modeled System Interface

**UserInput:**
- "call"
- "hang up"
- "cancel"

**SIPRequest:** "BYE"
**SIPResponse:** "180 Ringing",
"200 OK",
"486 Busy Here",
"487 Request Terminated"

**UserOutput:**
- "ringing"
- "call ended"
- "call established"
- "timeout"

**SIP User Agent Client**

**Network**

**SIPRequest:** "ACK", "BYE",
"CANCEL", "INVITE"
**SIPResponse:** "200 OK"
QML System Block and Message Definition

```java
system {
  Inbound  userIn  :  UserInput;
  Outbound userOut :  UserOutput;
  Inbound  netIn   :  SIPResponse, SIPRequest;
  Outbound netOut :  SIPResponse, SIPRequest;
}

record SIPRequest
{
  SIPRequestLine startLine;

  HeaderFieldCallId callId;
  HeaderFieldContact contact;
  HeaderFieldCSeq cSeq;
  HeaderFieldFrom from;
  HeaderFieldMaxForwards maxForwards;
  HeaderFieldTo to;
  HeaderFieldVia via;

  String msgBody;
}
```
The Statechart Diagram
Statechart Example: Call initiation

Paragraph numbers in comments and requirements refer to RFC 3261 SIP: Session Initiation Protocol.

Triggered by user action, send INVITE request. (We are initiating a new call.)

Init

Calling

Ringing

netIn:SIPResponse[msg.statusText.statusCode == 180]/
requireResponse(msg, theINVITE);
remoteTag = msg.to.tag;

netIn:SIPResponse[msg.statusText.statusCode == 486]/
requireResponse(msg, theINVITE);
Example for Java-like QML Action Language

• Implementation of action to send a SIP INVITE request:

```java
protected void sendInvite() {
    // initialize state variables
    this.localTag = "";
    this.remoteTag = "";
    // build SIP INVITE request with default values
    theINVITE = getRequestBase("INVITE", getSystemGeneratedValue());
    // store from tag for later use
    localTag = theINVITE.from.tag;
    // set contact header and message body values
    theINVITE.contact.address = "sip:" + getCallerSipUri();
    theINVITE.body = getSystemGeneratedValue();
    netOut.send(theINVITE);
}
```

• Method is referenced from statechart diagram

• System generated values are symbolic values which need to be managed at runtime by TTCN-3 harness
A Requirement in the Model

requirement "13.2.2.4 2XX Responses/UAC core establishes session with ACK";

netIn:SIPResponse
[msg.statusCode == 200 && msg.body == theINVITE.body] requireResponse(msg, theINVITE);
ack(msg);
Loading the Model

![Automated Test Design™](image)

Copyright © Conformiq Inc. and its subsidiaries. All rights reserved.
Generating Tests from Models
Results: Coverage Editor

[Image of a software window showing coverage goals for SIPClient - Eclipse Platform]
Results: Requirements Traceability Matrix

<table>
<thead>
<tr>
<th>Testing Goals</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.1.1.2 INVITE timers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.1.2.2 Non-INVITE timers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13.2.2.4 2xx Responses</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UAC core establishes session with ACV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15.1.1.1 terminating a session</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Control Flow</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conditional branching</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>State Chart</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Results: Test Case List

<table>
<thead>
<tr>
<th>Name</th>
<th>Created</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 branch: guard in SIPClient.Init-&gt;SIPClient.Calling-0:1 [0]</td>
<td>2009-10-26 03:15</td>
</tr>
<tr>
<td>4 requirement: 17.1.1.2 INVITE timers/Resends INVITE after A timeout</td>
<td>2009-10-26 03:15</td>
</tr>
<tr>
<td>5 requirement: 17.1.1.2 INVITE timers/terminates INVITE cycle after F timeout</td>
<td>2009-10-26 03:15</td>
</tr>
<tr>
<td>9 requirement: 17.1.2.2 Non-INVITE timers/Resends CANCEL after E timeout</td>
<td>2009-10-26 03:15</td>
</tr>
<tr>
<td>11 requirement: 17.1.2.2 Non-INVITE timers/Terminates CANCEL cycle after F timeout</td>
<td>2009-10-26 03:15</td>
</tr>
<tr>
<td>12 requirement: 13.2.5.4 2xx Responses/UAC core establishes session with ACK</td>
<td>2009-10-26 03:15</td>
</tr>
<tr>
<td>13 requirement: 15.1 Terminating a session/UAC core sends OK in response to BYE</td>
<td>2009-10-26 03:15</td>
</tr>
<tr>
<td>14 requirement: 15.1 Terminating a session/UAC core terminates a session by sending BYE</td>
<td>2009-10-26 03:15</td>
</tr>
<tr>
<td>16 requirement: 17.1.2.2 Non-INVITE timers/Resends BYE after E timeout</td>
<td>2009-10-26 03:15</td>
</tr>
<tr>
<td>17 requirement: 17.1.2.2 Non-INVITE timers/Terminates BYE cycle after F timeout</td>
<td>2009-10-26 03:15</td>
</tr>
</tbody>
</table>
Results: Abstract Test Case View
Results: Test Steps and Test Data

<table>
<thead>
<tr>
<th>Message / Field</th>
<th>Port / Field value</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>UserInput</td>
<td>to userIn</td>
<td>0.0</td>
</tr>
<tr>
<td>cmd</td>
<td>&quot;invite&quot;</td>
<td></td>
</tr>
<tr>
<td>SipRequest</td>
<td>from netOut</td>
<td>0.0</td>
</tr>
<tr>
<td>startLine</td>
<td></td>
<td></td>
</tr>
<tr>
<td>method</td>
<td>&quot;INVITE&quot;</td>
<td></td>
</tr>
<tr>
<td>requestURI</td>
<td>&quot;sip:160@127.0.0.1:5060&quot;</td>
<td></td>
</tr>
<tr>
<td>callID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>contact</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSeq</td>
<td></td>
<td></td>
</tr>
<tr>
<td>from</td>
<td></td>
<td></td>
</tr>
<tr>
<td>maxForwards</td>
<td></td>
<td></td>
</tr>
<tr>
<td>to</td>
<td></td>
<td></td>
</tr>
<tr>
<td>via</td>
<td></td>
<td></td>
</tr>
<tr>
<td>body</td>
<td>&quot;#SYSTEM_GENERATED_5_&quot;</td>
<td></td>
</tr>
<tr>
<td>SipResponse</td>
<td>to netIn</td>
<td>0.0</td>
</tr>
<tr>
<td>statusLine</td>
<td></td>
<td></td>
</tr>
<tr>
<td>statusCode</td>
<td>180</td>
<td></td>
</tr>
<tr>
<td>callID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>contact</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CSeq</td>
<td></td>
<td></td>
</tr>
<tr>
<td>from</td>
<td></td>
<td></td>
</tr>
<tr>
<td>to</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Results: Test Case Dependency Matrix

![Test Case Dependency Matrix](image-url)

<table>
<thead>
<tr>
<th>Prerequisite</th>
<th>Dependent</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1: branch: guard in SIPClient.Init -&gt; SIPClient</td>
<td>1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17</td>
</tr>
<tr>
<td>#2: branch: guard in SIPClient.Calling -&gt; SIPClient</td>
<td></td>
</tr>
<tr>
<td>#3: branch: guard in SIPClient.Ringing -&gt; SIPClient</td>
<td></td>
</tr>
<tr>
<td>#4: requirement: 17.1.1.2 INVITE timers</td>
<td></td>
</tr>
<tr>
<td>#5: requirement: 17.1.1.2 INVITE timers</td>
<td></td>
</tr>
<tr>
<td>#6: branch: guard in SIPClient.Ringing -&gt; SIPClient</td>
<td></td>
</tr>
<tr>
<td>#7: branch: guard in SIPClient.Cancelling</td>
<td></td>
</tr>
<tr>
<td>#8: branch: guard in SIPClient.Waiting Resp</td>
<td></td>
</tr>
<tr>
<td>#9: requirement: 17.1.2.2 Non-INVITE timers</td>
<td></td>
</tr>
<tr>
<td>#10: branch: else branch in SIPClient</td>
<td></td>
</tr>
<tr>
<td>#11: requirement: 17.1.2.2 Non-INVITE timers</td>
<td></td>
</tr>
<tr>
<td>#12: requirement: 13.2.2.4 2xx Resp</td>
<td></td>
</tr>
<tr>
<td>#13: requirement: 15.1 Terminating a session</td>
<td></td>
</tr>
<tr>
<td>#14: requirement: 15.1 Terminating a session</td>
<td></td>
</tr>
<tr>
<td>#15: branch: guard in SIPClient.Terminating</td>
<td></td>
</tr>
<tr>
<td>#16: requirement: 17.1.2.2 Non-INVITE timers</td>
<td></td>
</tr>
<tr>
<td>#17: requirement: 17.1.2.2 Non-INVITE timers</td>
<td></td>
</tr>
</tbody>
</table>

---

Copyright © Conformiq Inc. and its subsidiaries. All rights reserved. Tuesday, May 11, 2010
Rendering the Tests as TTCN-3
testcase tc_12()
runs on CQ_MTC system MyTSI
{
  var float v_last_timeout := 0.0;
  var default v_cq_default_ref;

  f_start_test_case();
  v_cq_default_ref := activate(a_cq_default());
  f_send_UserInput_to_userIn(m_UserInput92);
  t_cq_timer.start((0.0 - v_last_timeout) +
    f_receive_SIPRequest_from_netOut(m_expected);
  t_cq_timer.stop; v_lastTimeout := 0.0;
  f_send_SIPResponse_to_netIn(m_SIPResponse94);
  t_cq_timer.start((0.0 - v_last_timeout) +
    f_receive_UserOutput_from_userOut(m_expecte;
  t_cq_timer.stop; v_lastTimeout := 0.0;
  f_send_SIPResponse_to_netIn(c_SIPResponse9;
  t_cq_timer.start((0.0 - v_last_timeout) +
    f_receive_SIPRequest_from_netOut(m_expect;
  t_cq_timer.stop; v_lastTimeout := 0.0;
  log(“requirement: 13.2.2.4 2xx Response/UA 
...
TTCN-3 Test Data

Template:
```
template SIPRequest m_SIPRequest93 :=
{
    startLine := {
        method := "INVITE",
        requestURI := "sip:100@127.0.0.1:5061"
    },
    callId := {
        callId := "#SYSTEM_GENERATED_1_"
    },
    contact := {
        addr := "sip:150@127.0.0.1:5062"
    },
    cSeq := {
        sequenceNumber := "#SYSTEM_GENERATED_2_"
    },
    from_ := {
        addr := "sip:150@127.0.0.1:5061",
        tag := "#SYSTEM_GENERATED_3_"
    },
    maxForwards := 70,
}
```
Conclusions

• **System model driven Automated Test Design** offers significant gains in **productivity**
  – Faster test development and improved test quality
  – Wider test coverage and guaranteed requirement coverage
  – Cost-effective maintenance
  – Earlier test validation & detection of specification defects
  – Independence from test execution environments

• By combining **Conformiq Designer** with **TTCN-3** you get the **best of both worlds**:
  – All the benefits of Automated Test Design
  – A well-defined and standardized environment for test execution
Contact Information

Jani Koivulainen
Director, Customer Success

jani.koivulainen@conformiq.com
+358408654351
http://www.conformiq.com