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 Ideabytes

Innovation is Business

Tutorial Outline

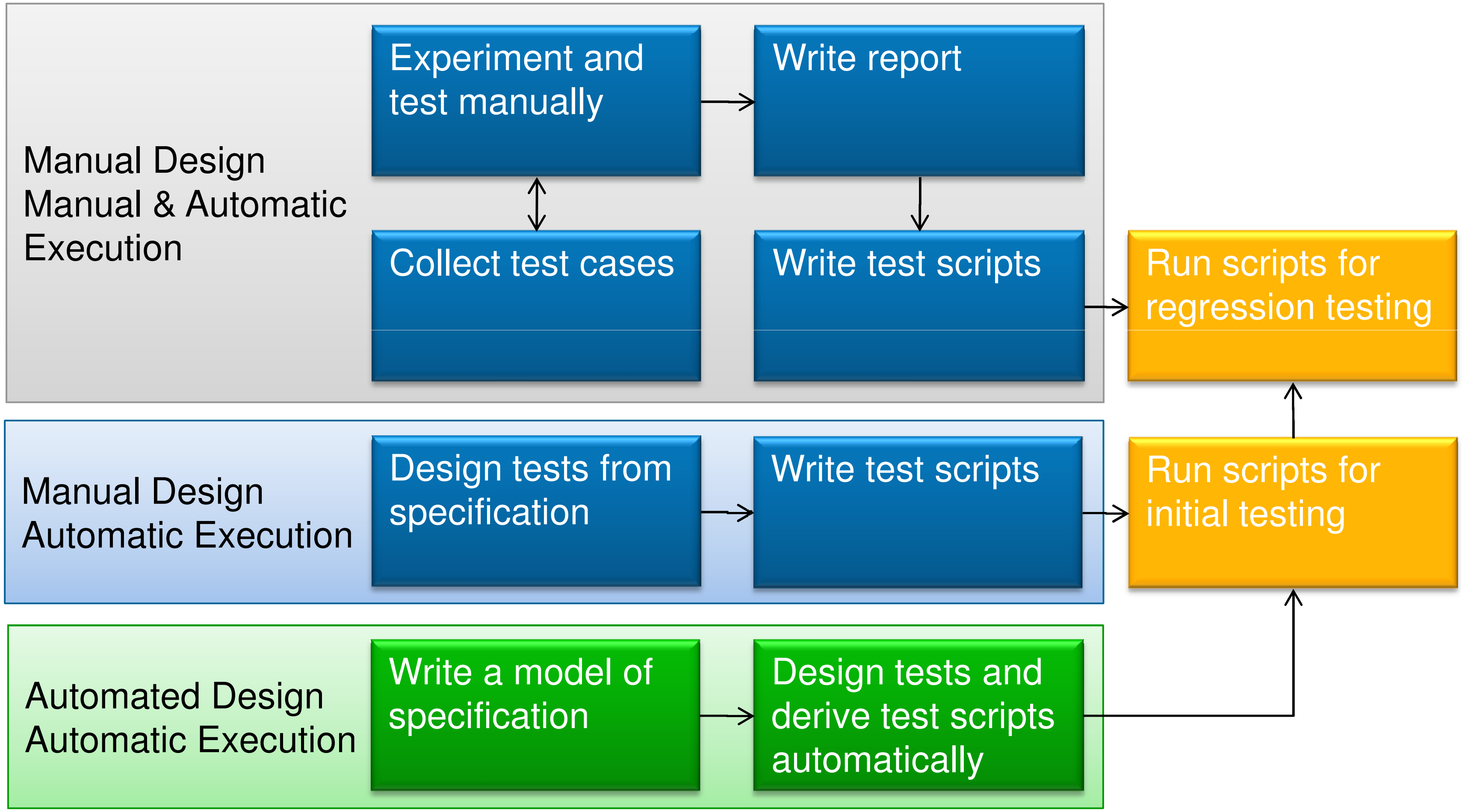
- Automated Test Design
- Why Automated Test Design?
- Conformiq Designer
- Conformiq Designer and TTCN-3
- SIP Example Walkthrough

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

Write report

Collect test cases

Write test scripts

Run scripts for regression testing

Manual Design
Automatic Execution

Design tests from specification

Write test scripts

Run scripts for initial testing

Automated Design
Automatic Execution

Write a model of specification

Design tests and derive test scripts automatically

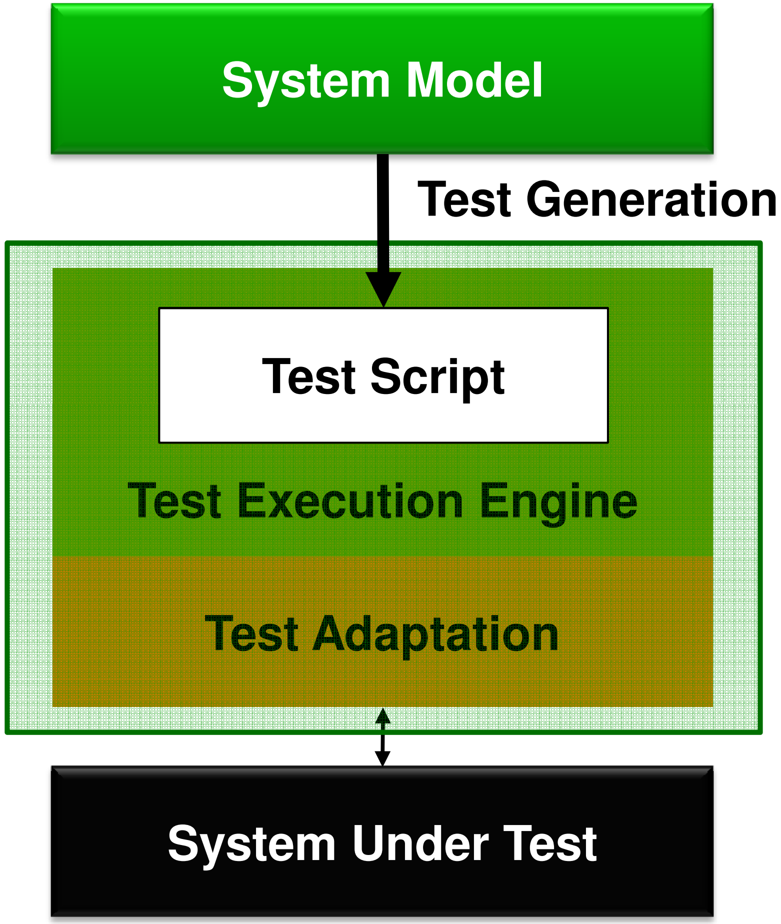
What is Model-Based Testing?

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

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 specificatoin defects
  - Independence from test execution environment

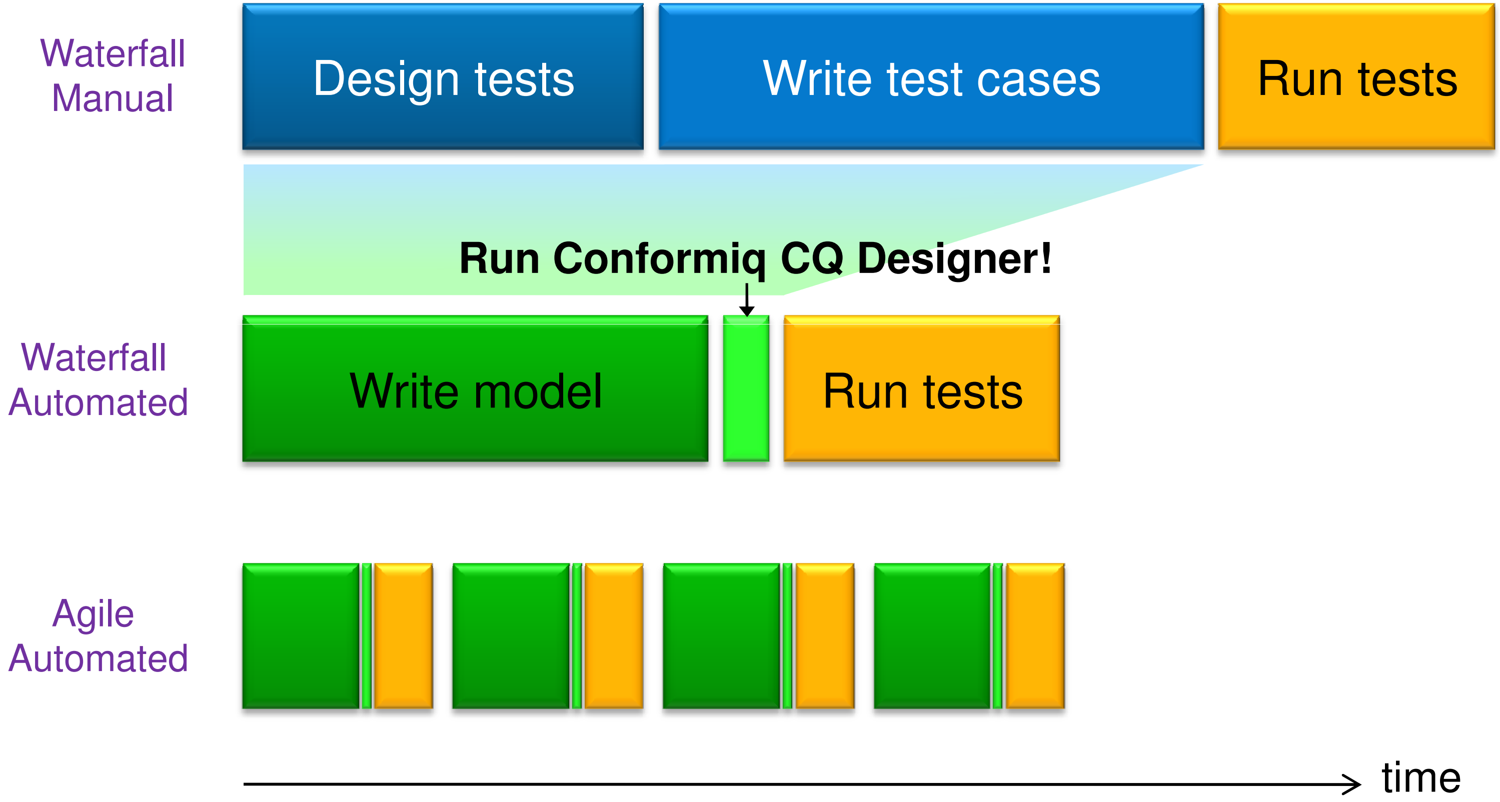
Integration of Automated Test Design



Test execution engine and adaptation can be reused “as is”!

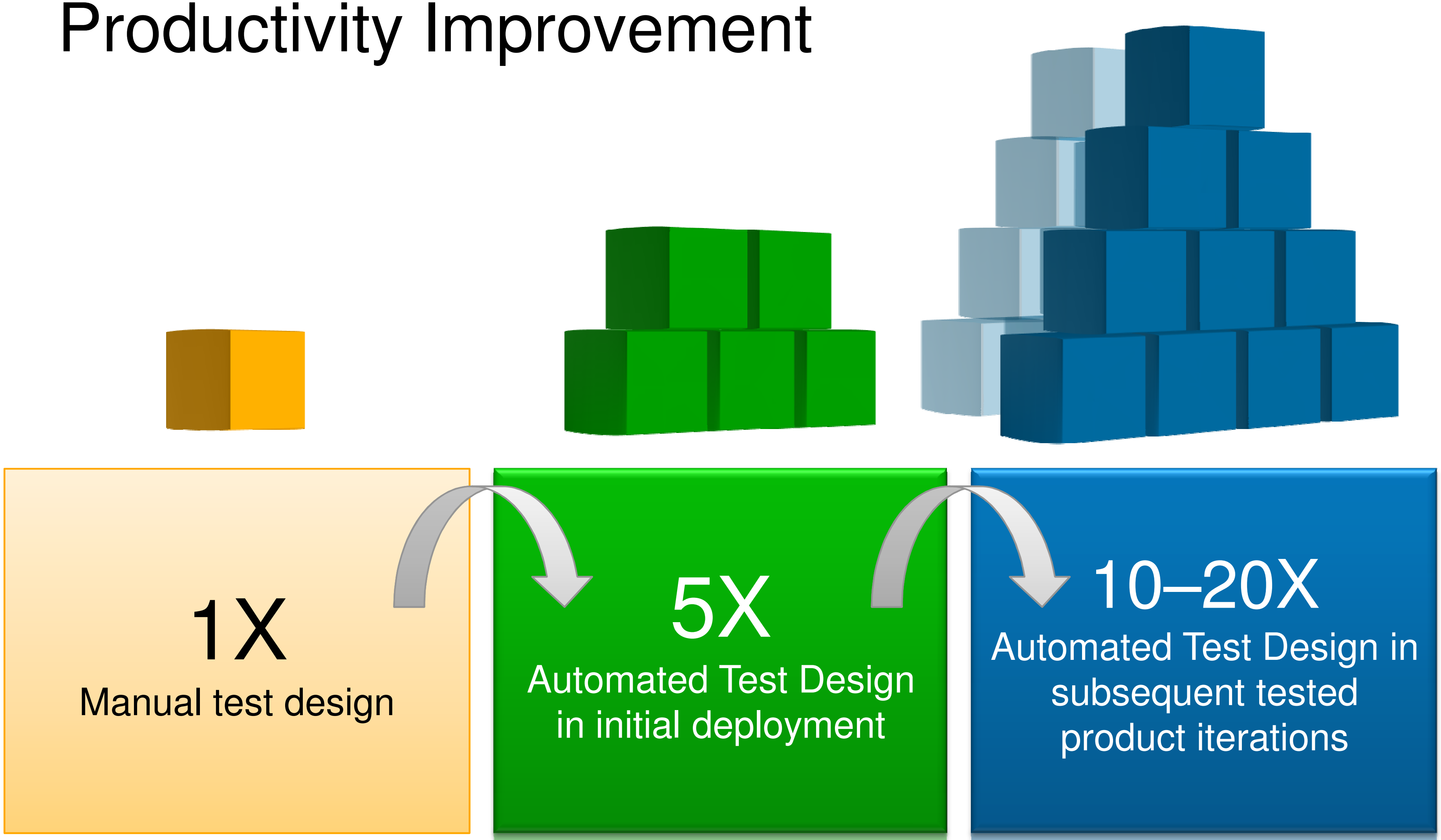
Why Automated Test Design?

Manual vs. Automated Test Design



Find and Fix more defects sooner

Productivity Improvement



1X

Manual test design

5X

Automated Test Design in initial deployment

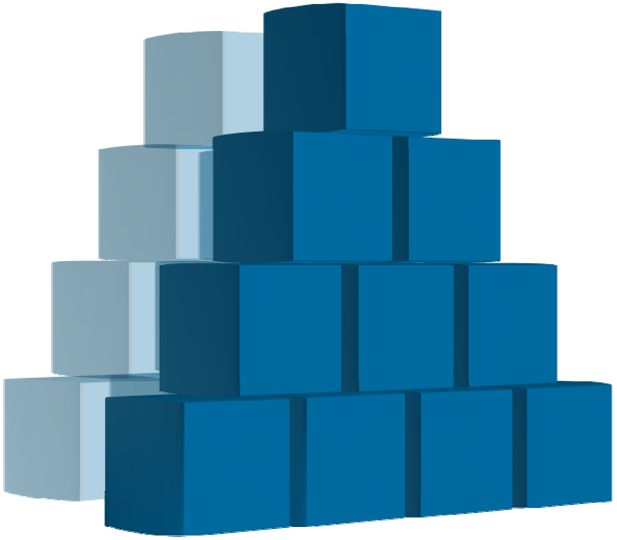
10–20X

Automated Test Design in subsequent tested product iterations

Source: average results from customer benchmarks

10–20X Elaborated

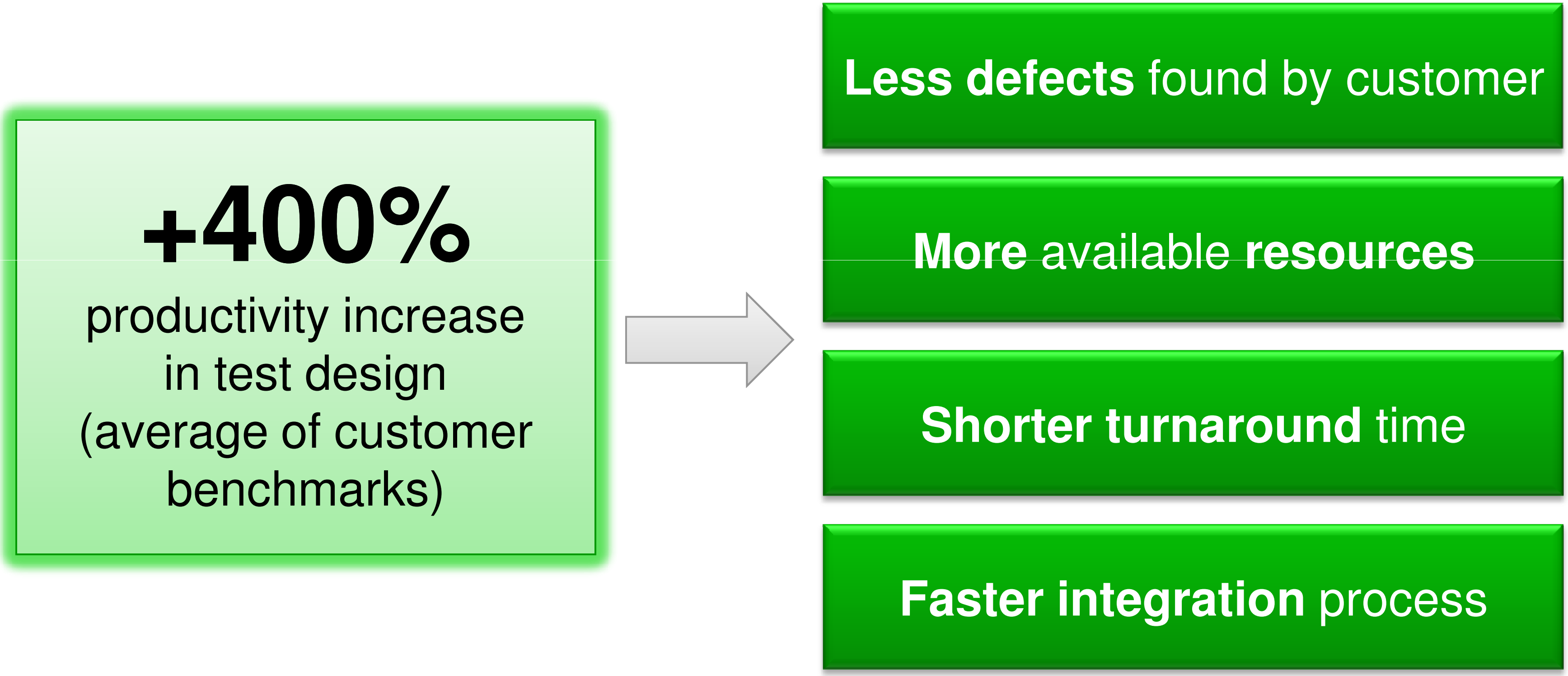
- Higher test **quality**
- The same model is the source of all tests
  → **easier to maintain**
- Models are **easier to review** and communicate than test scripts
- Model components are easier to **reuse**, **share** and **compose** than test cases which are ”snapshots”



10–20X

Automated Test Design in subsequent tested product iterations

Productivity Improvement as an Enabler



+400%

productivity increase in test design (average of customer benchmarks)

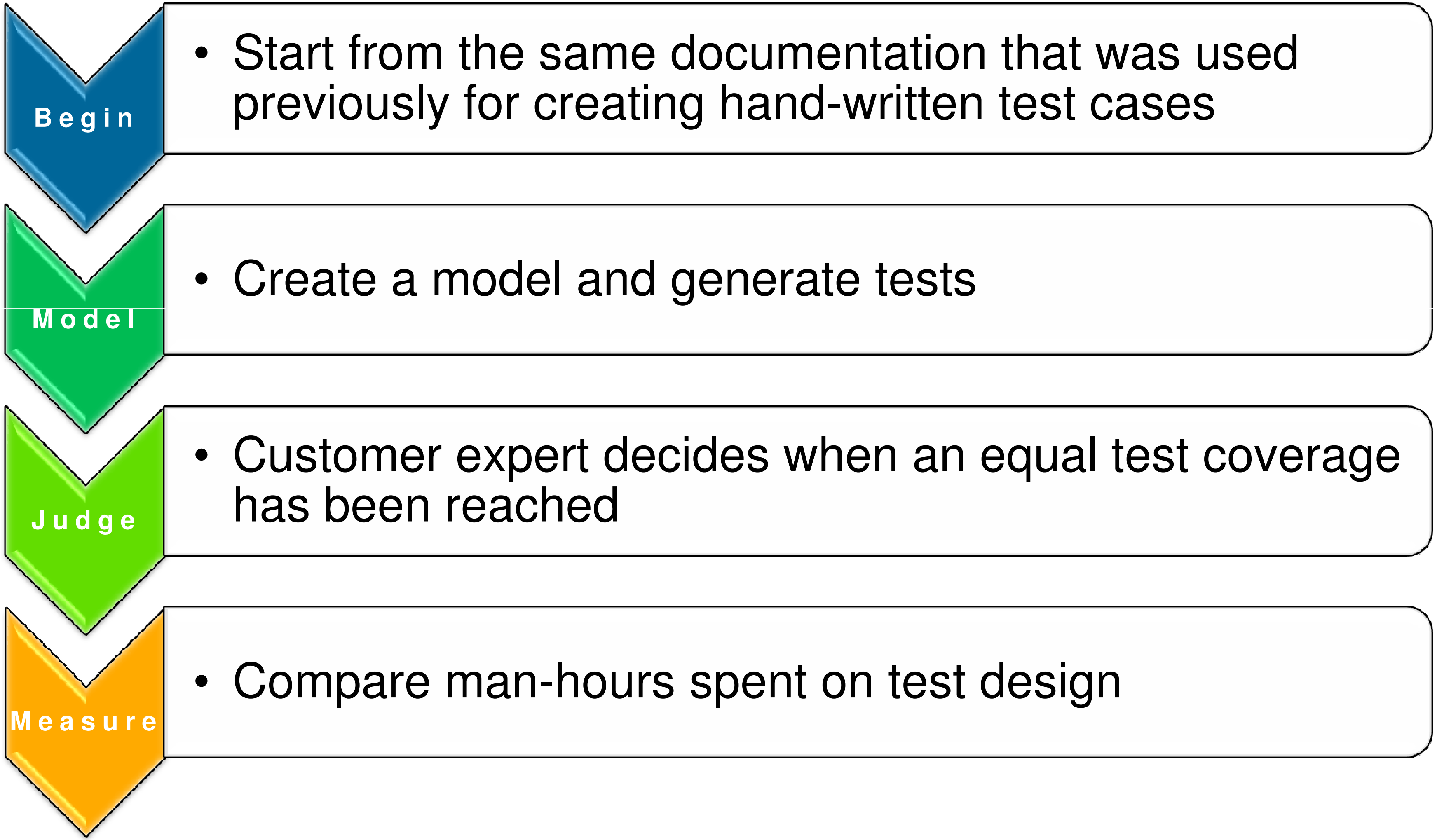
Less defects found by customer

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

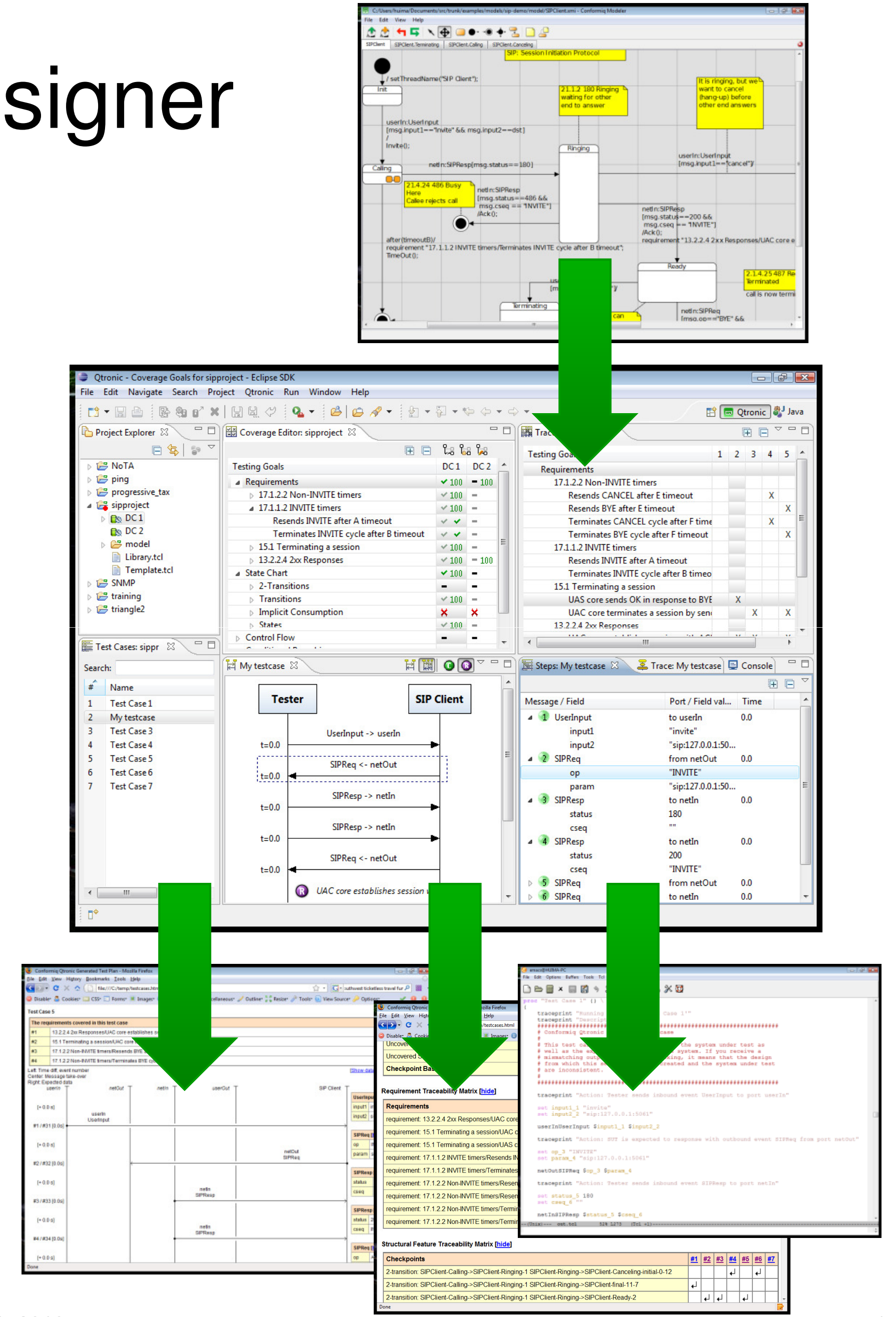
Measure

- Compare man-hours spent on test design

Conformiq Designer

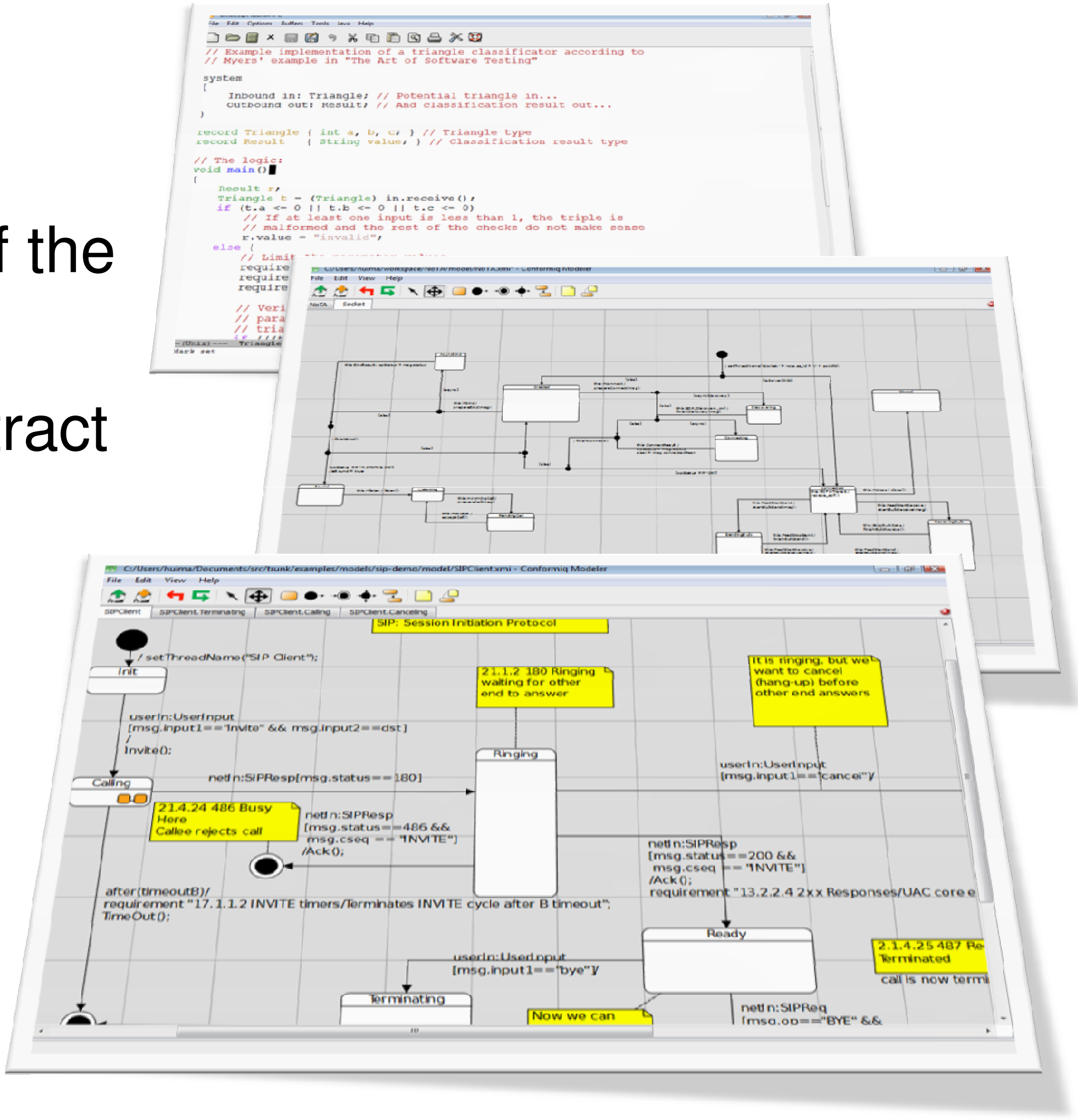
ATD with Conformiq Designer

- 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


SIP: Session Initiation Protocol
Qtronic - Coverage Goals for sipproject - Eclipse SDK
Project Explorer
Coverage Editor: sipproject
Testing Goals
Requirements
Test Cases: sippr
My testcase
Tester
SIP Client
Steps: My testcase
Trace: My testcase
Console
Requirement Traceability Matrix
Structural Feature Traceability Matrix

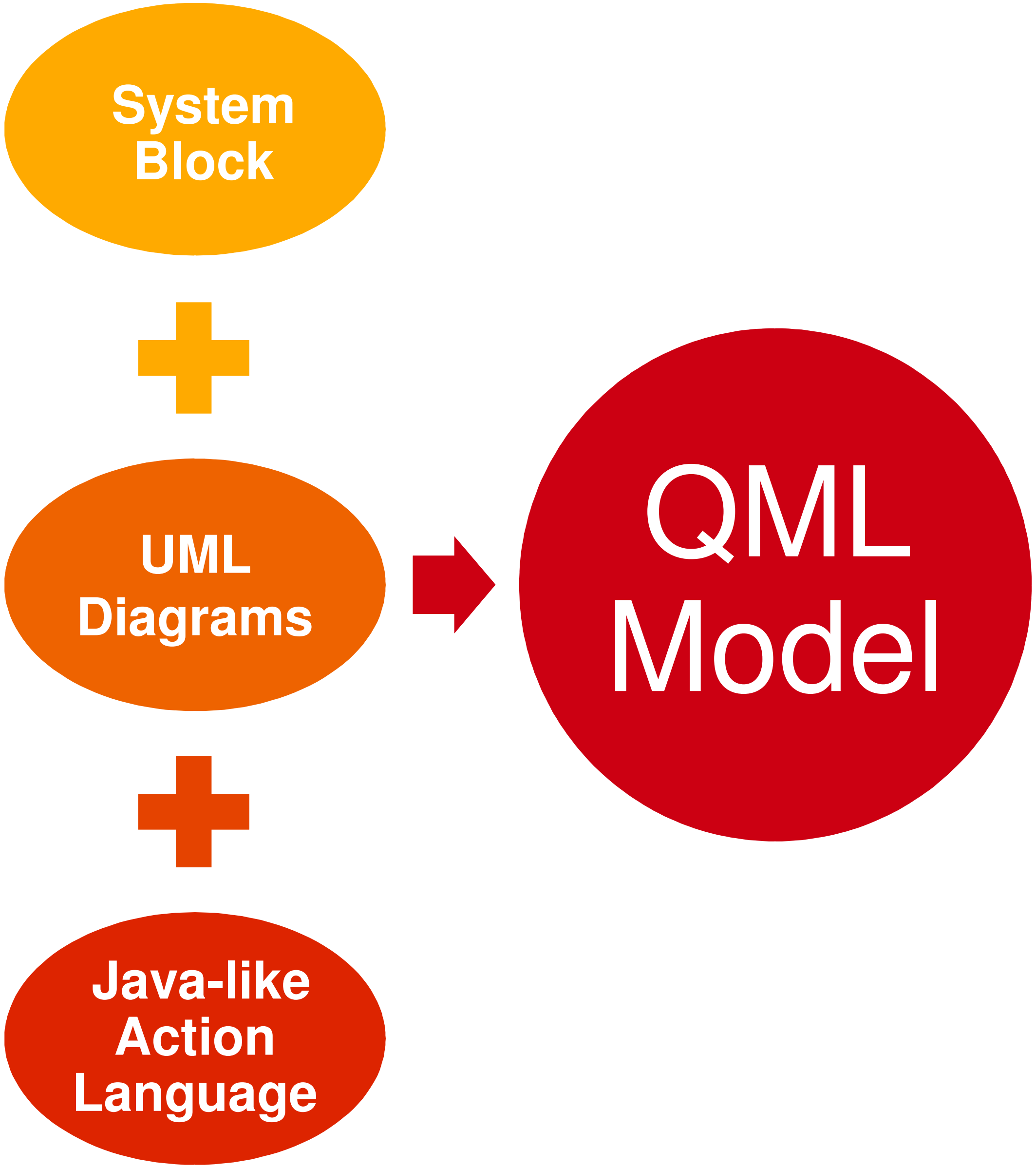
The System Model…

- 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

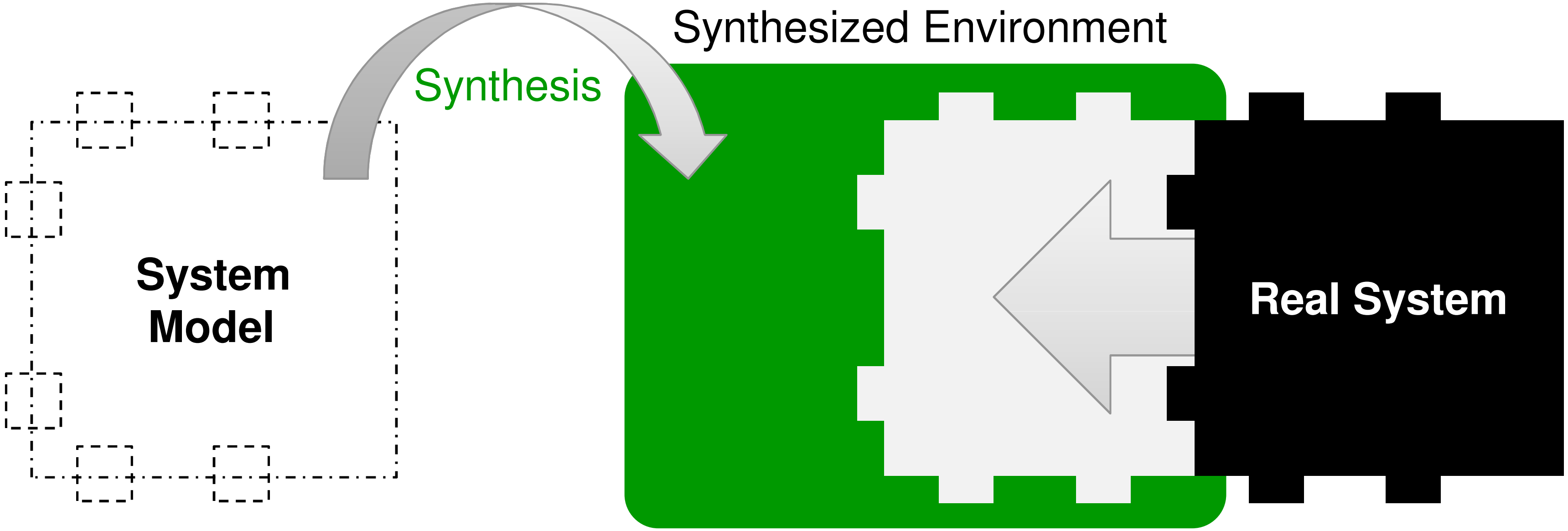


```
// Example implementation of a triangle classificator according to
// Myers' example in "The Art of Software Testing"

system
{
    Inbound in: Triangle; // Potential triangle in...
    Outbound out: Result; // And classification result out...
}

record Triangle { int a, b, c; } // Triangle type
record Result   { String value; } // Classification result type

// The logic:
void main()
{
    Result r;
    Triangle t = (Triangle) in.receive();
    if (t.a <= 0 || t.b <= 0 || t.c <= 0)
        // If at least one input is less than 1, the triple is
        // malformed and the rest of the checks do not make sense
        r.value = "invalid";
    else {
        // Limit
        require
        require
        require

        // Veri
        // para
        // tria
```

Conformiq Modeling Language (QML)



From the System Model to the ”Black Box”



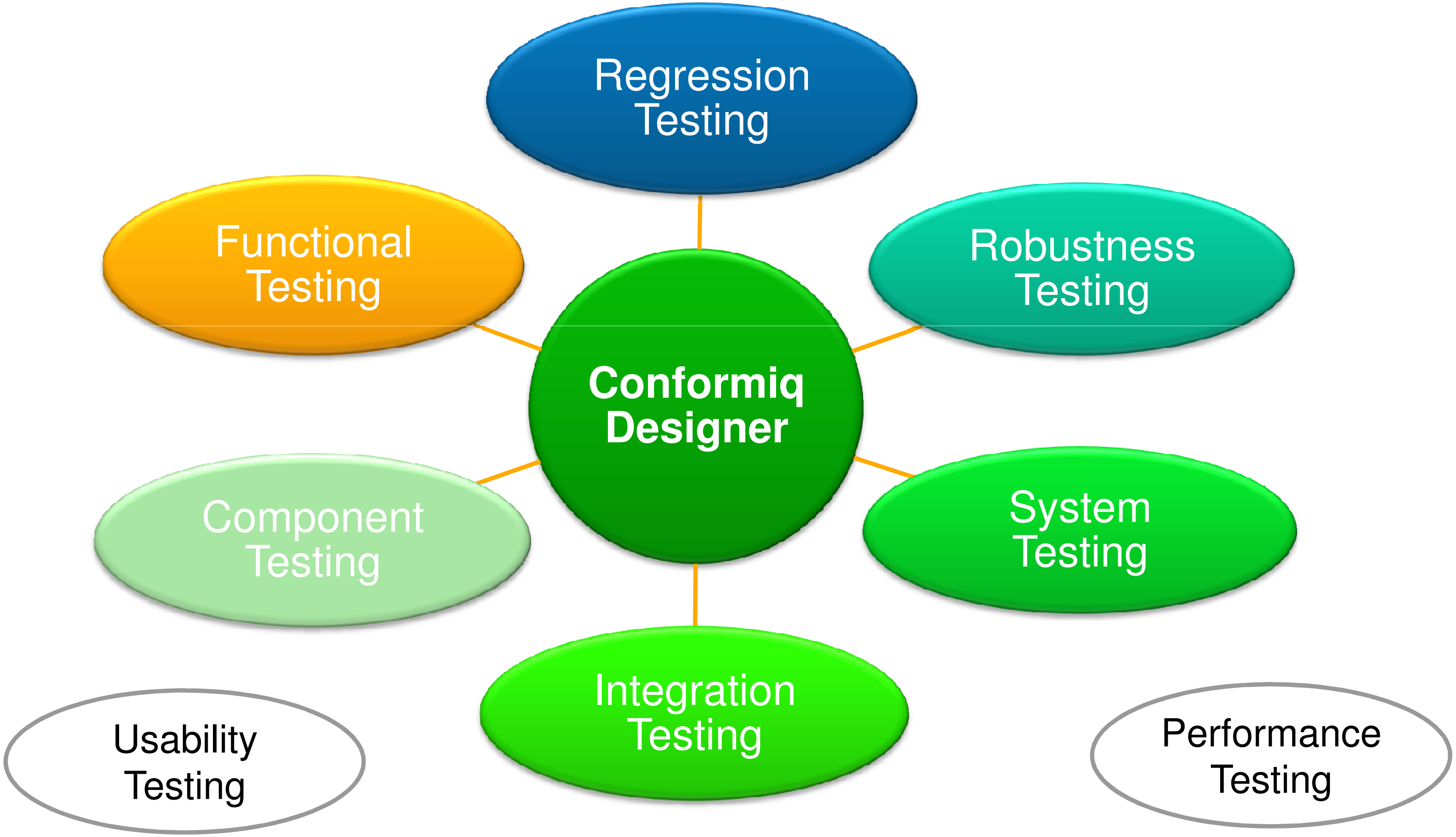
Coverage Criteria supported by CQ Designer

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

Conformiq Designer Features

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

Conformiq Designer Applicability



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

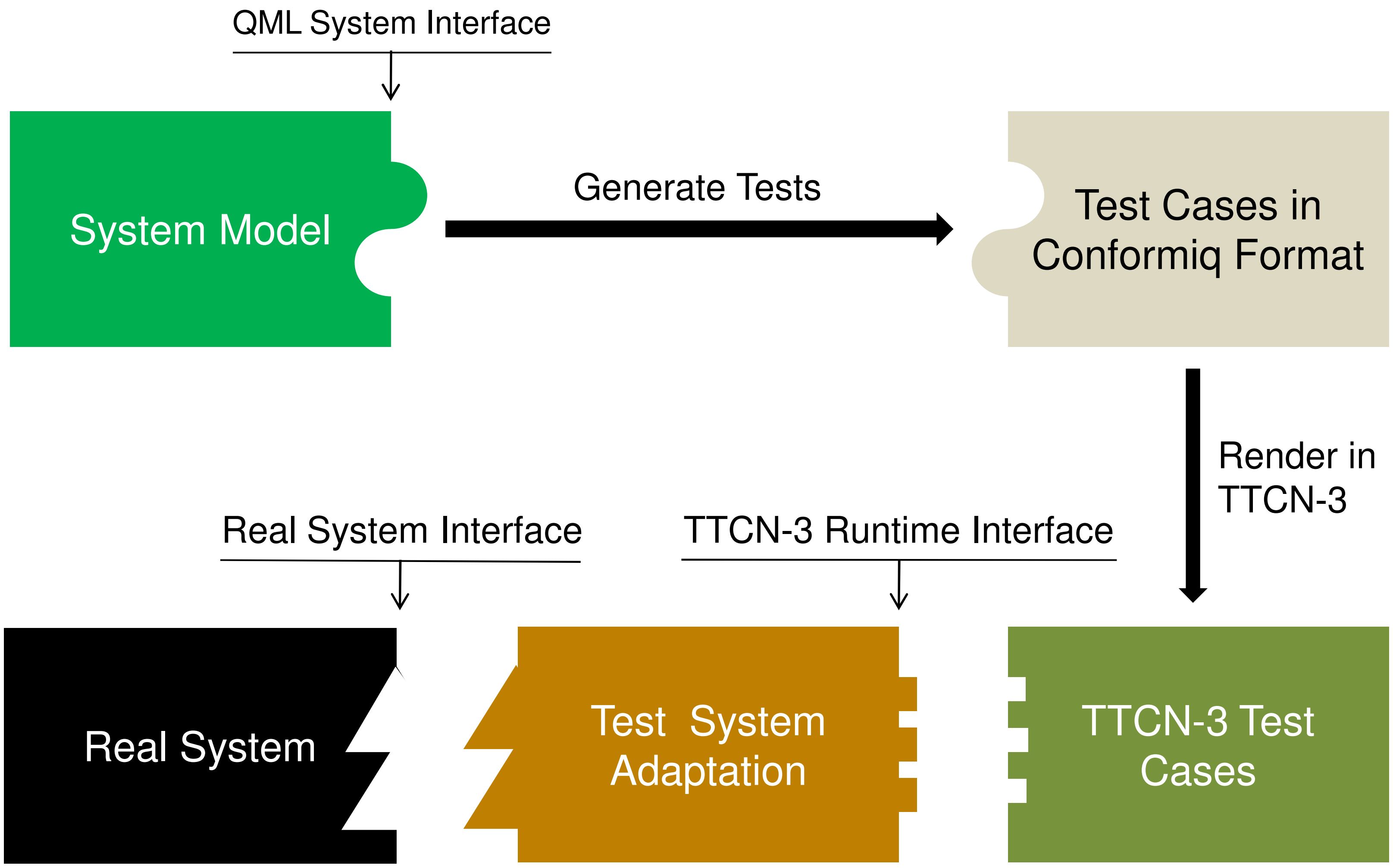
Experiences with TTCN-3 Tools

- **MessageMagic** (Elvior)
- **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?

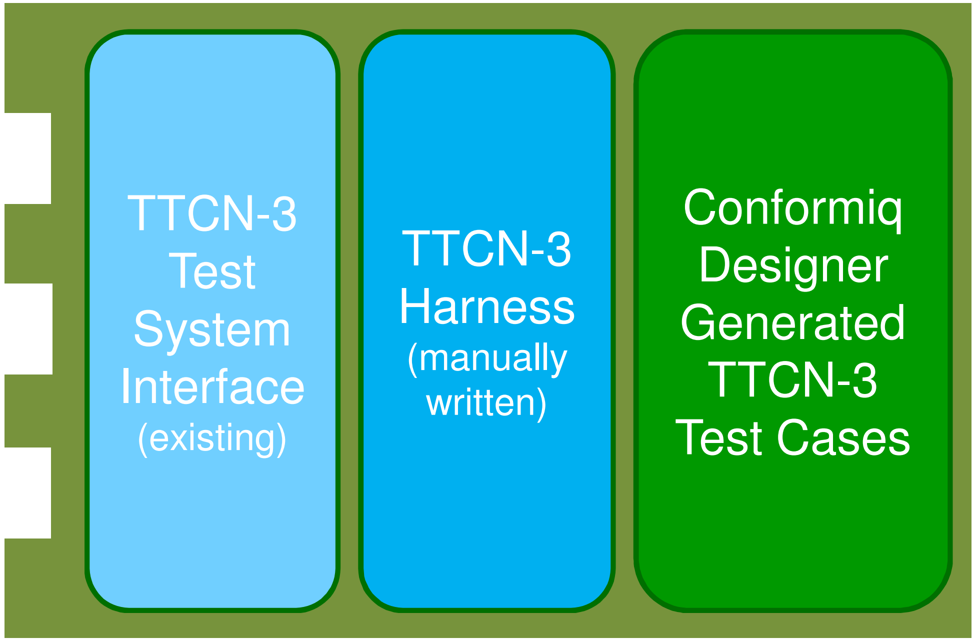
- Benefits for TTCN-3 users:
  - Automated test case design, writing, and documentation
  - Consistent test design and quality
  - Guaranteed requirement coverage
  - More efficient test suite maintenance
  - Easier test artefact review, reuse and sharing
  - Reuse existing test execution platforms
-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

Test Adaptation


TTCN-3 Test System Interface (existing)
TTCN-3 Harness (manually written)
Conformiq Designer Generated TTCN-3 Test Cases

A Test Harness Implementation Example

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

```
f_send_SIPRequest_to_netIn( c_SIPRequest15 );
```

- The function is implemented by the TTCN-3 Harness

```
// Finalize and transform SIP request to TTCN-3 type
// structure if different from the one used by model
function f_send_SIPRequest_to_netIn( in SIPRequest p_req )
runs on CQ_MTC
{
  var TTCN3_SIP_Request v_TTCNReq;
  v_TTCNReq := f_prepare_send_SIPRequest( p_req );
  netIn.send( v_TTCNReq );
}
```

Test Harness - Preparation before Sending

```
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

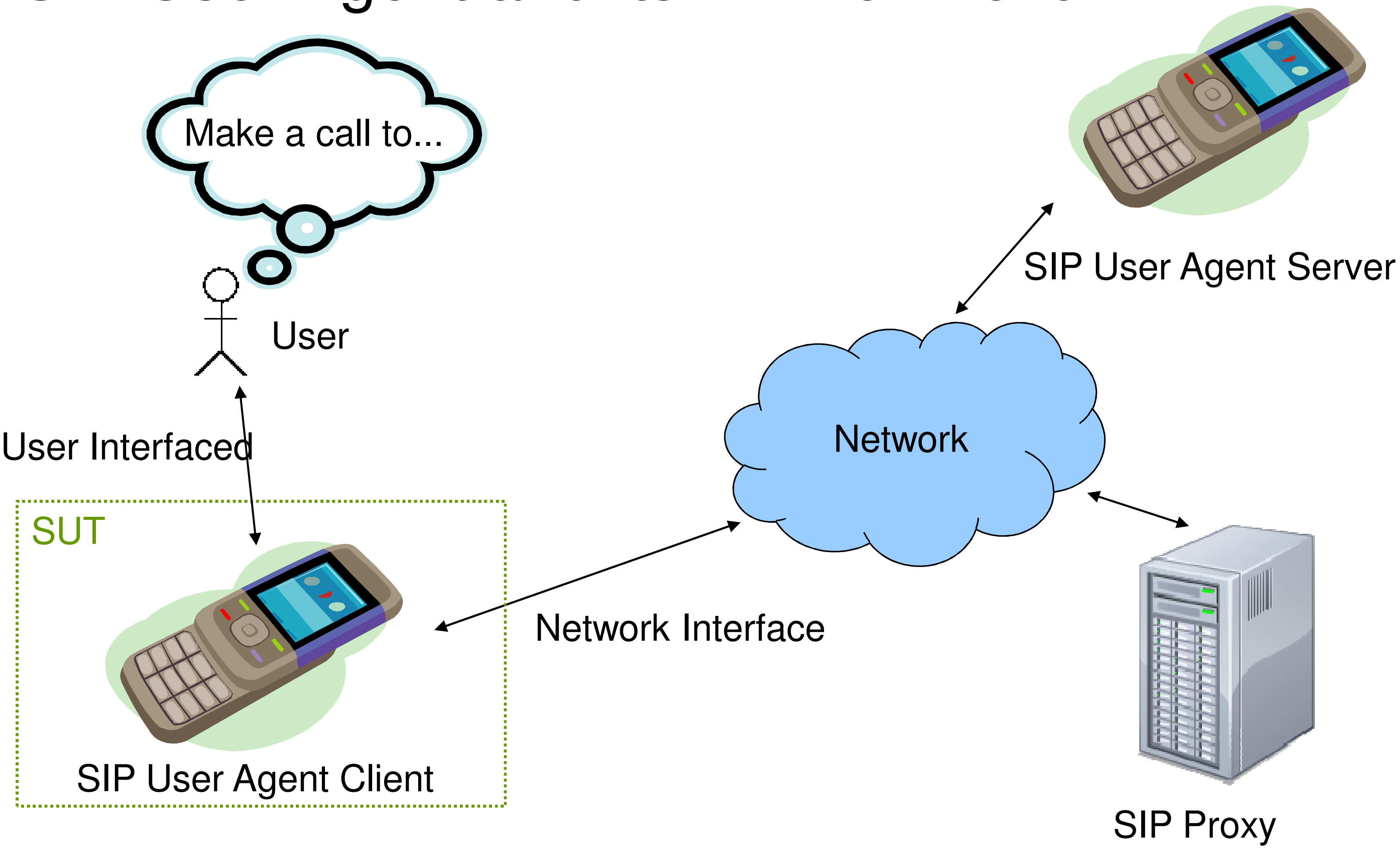
```
function f_prepare_and_match_SIPRequest ( in SIPRequest p_expReq,
                                in TTCN3_SIP_Request p_rcvTTCNReq )
runs on CQ_MTC return SIPRequest
{
   // 1. transform from TTCN-3 to abstract type structure if needed
   var SIPRequest v_rcvReq := f_SIPRequest_transform2cq( p_rcvTTCNReq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

   // 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



Tested Functionality

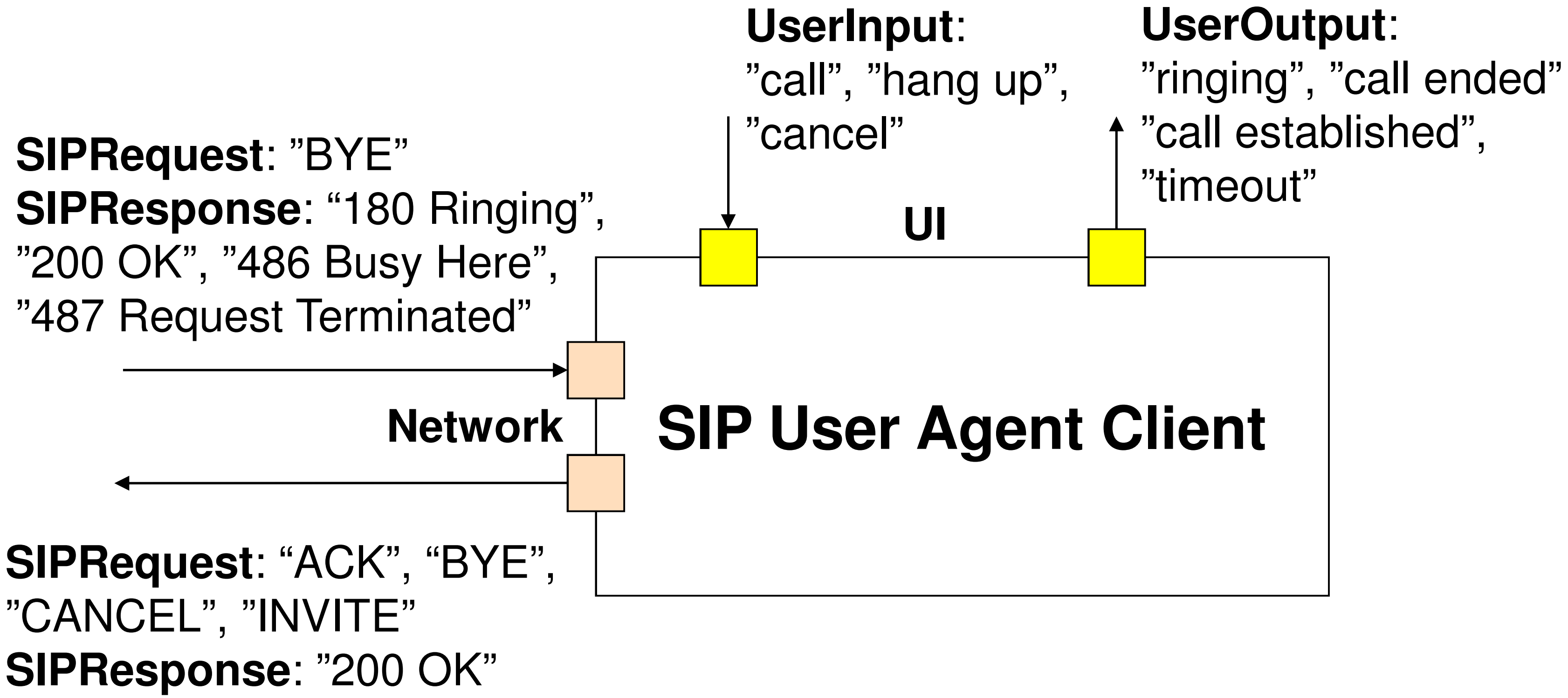
- Call establishment (“SIP INVITE”)
- Call termination (“SIP BYE”)
  - caller-initiated
  - callee-initiated
- Call cancelation (“SIP CANCEL”)
- Timers
  - re-transmission
  - transaction

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”

UserOutput: ”ringing”, ”call ended” ”call established”, ”timeout”

UI

SIPRequest: ”BYE”
SIPResponse: “180 Ringing”, ”200 OK”, ”486 Busy Here”, ”487 Request Terminated”

Network

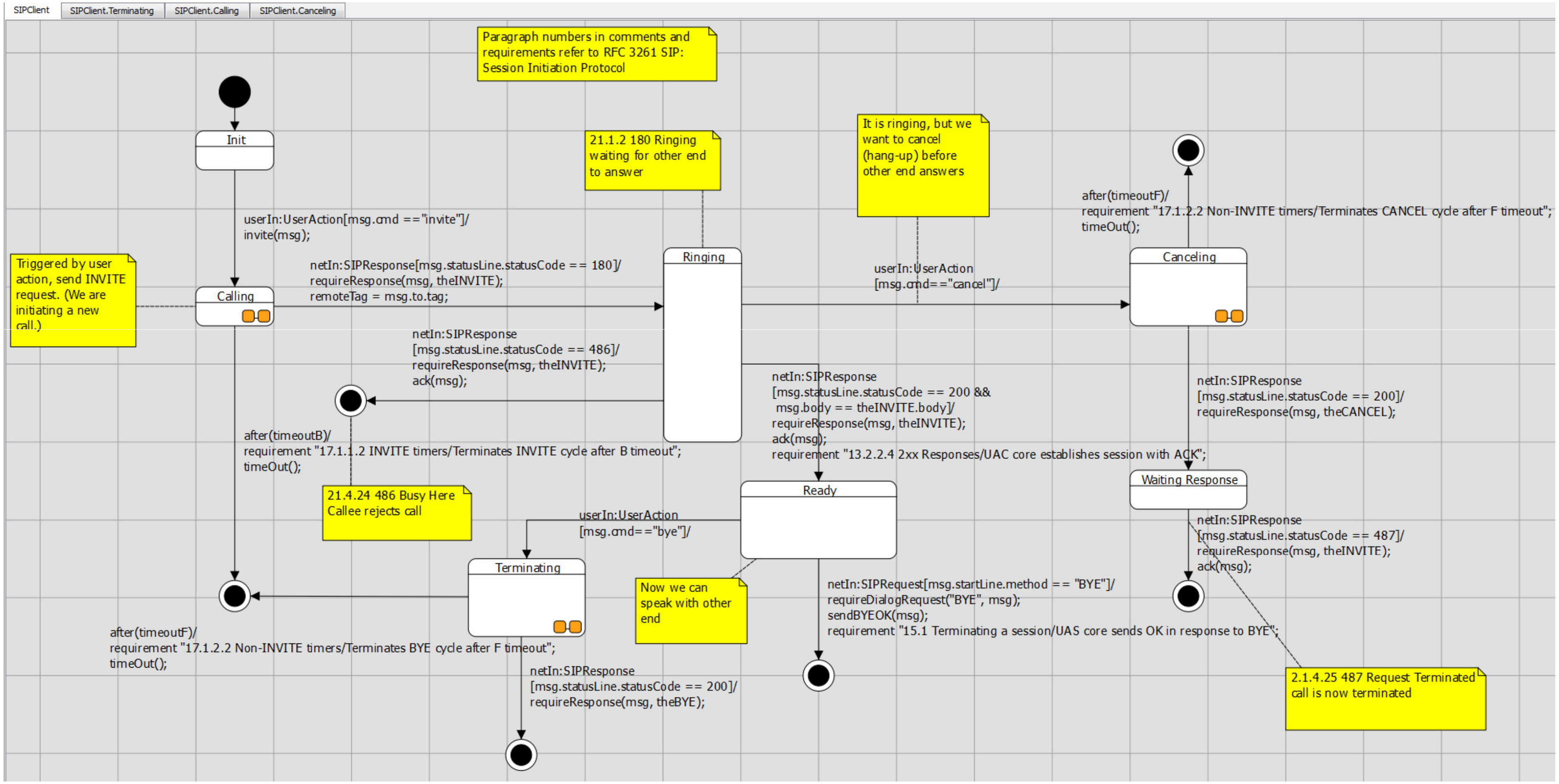
SIP User Agent Client

SIPRequest: “ACK”, “BYE”, ”CANCEL”, ”INVITE”
SIPResponse: ”200 OK”

QML System Block and Message Definition

```
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



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

21.1.2 180 Ringing waiting for other end to answer

It is ringing, but we want to cancel (hang-up) before other end answers

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

21.4.24 486 Busy Here Callee rejects call

Now we can speak with other end

2.1.4.25 487 Request Terminated call is now terminated

```
userIn:UserAction[msg.cmd =="invite"]/
invite(msg);
```

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

```
netIn:SIPResponse
[msg.statusLine.statusCode == 486]/
requireResponse(msg, theINVITE);
ack(msg);
```

```
after(timeoutB)/
requirement "17.1.1.2 INVITE timers/Terminates INVITE cycle after B timeout";
timeOut();
```

```
userIn:UserAction
[msg.cmd=="cancel"]/
```

```
after(timeoutF)/
requirement "17.1.2.2 Non-INVITE timers/Terminates CANCEL cycle after F timeout";
timeOut();
```

```
netIn:SIPResponse
[msg.statusLine.statusCode == 200 &&
 msg.body == theINVITE.body]/
requireResponse(msg, theINVITE);
ack(msg);
requirement "13.2.2.4 2xx Responses/UAC core establishes session with ACK";
```

```
netIn:SIPResponse
[msg.statusLine.statusCode == 200]/
requireResponse(msg, theCANCEL);
```

```
userIn:UserAction
[msg.cmd=="bye"]/
```

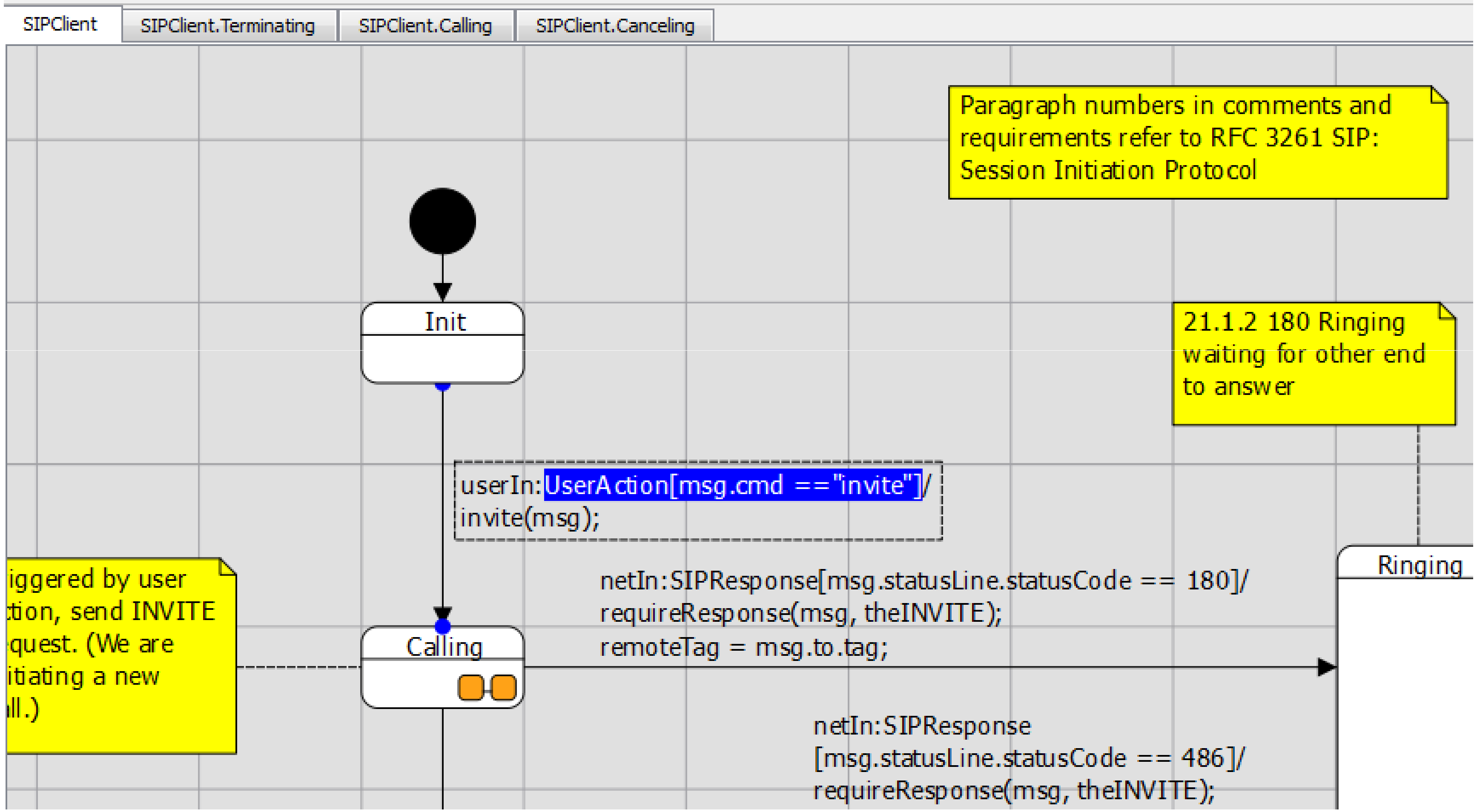
```
netIn:SIPResponse
[msg.statusLine.statusCode == 487]/
requireResponse(msg, theINVITE);
ack(msg);
```

```
netIn:SIPRequest[msg.startLine.method == "BYE"]/
requireDialogRequest("BYE", msg);
sendBYEOK(msg);
requirement "15.1 Terminating a session/UAS core sends OK in response to BYE";
```

```
after(timeoutF)/
requirement "17.1.2.2 Non-INVITE timers/Terminates BYE cycle after F timeout";
timeOut();
```

```
netIn:SIPResponse
[msg.statusLine.statusCode == 200]/
requireResponse(msg, theBYE);
```

Statechart Example: Call initiation



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

21.1.2 180 Ringing waiting for other end to answer

iggered by user tion, send INVITE quest. (We are itiating a new ll.)

```
userIn:UserAction[msg.cmd =="invite"]/
invite(msg);
```

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

```
netIn:SIPResponse
[msg.statusLine.statusCode == 486]/
requireResponse(msg, theINVITE);
```

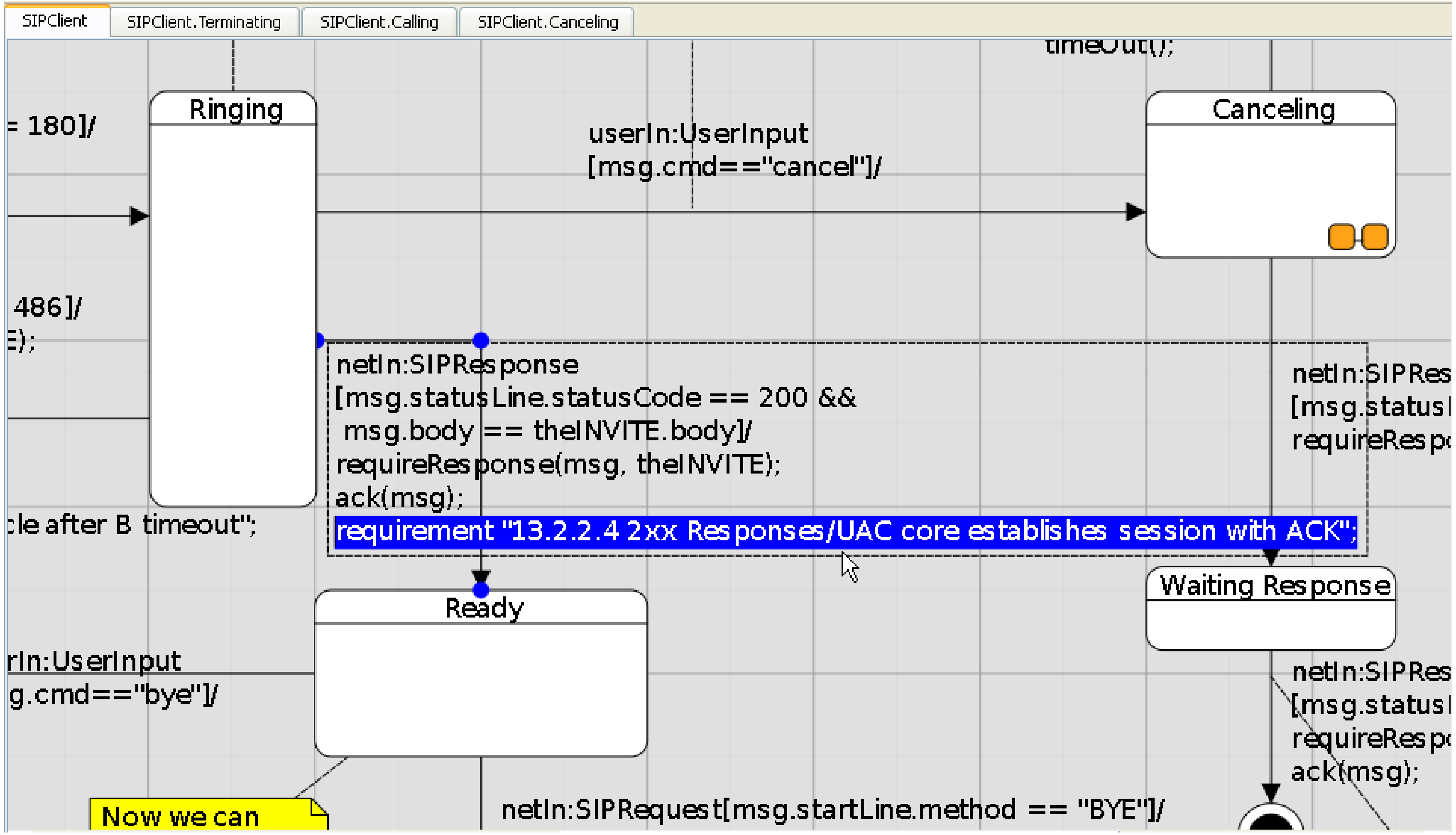
Example for Java-like QML Action Language

- Implementation of action to send a SIP INVITE request:

```
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);
}
```

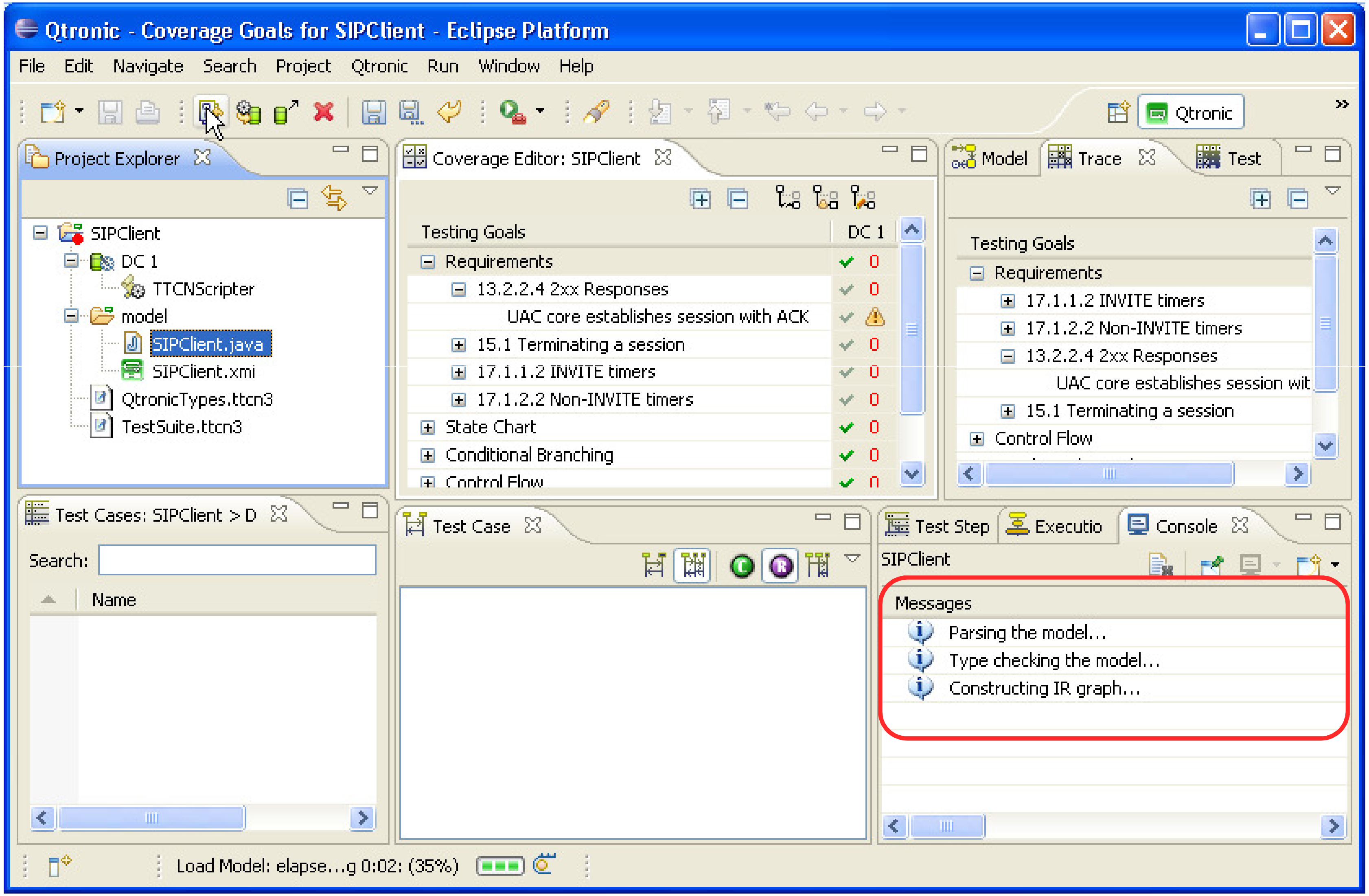
- Method is referenced from statechart diagram
- System generated values are symbolic values which need to be managed at runtime by TTCN-3 harness

A Requirement in the Model



```
timeOut();
= 180]/
userIn:UserInput
[msg.cmd=="cancel"]/
486]/
E);
netIn:SIPResponse
[msg.statusLine.statusCode == 200 &&
 msg.body == theINVITE.body]/
requireResponse(msg, theINVITE);
ack(msg);
requirement "13.2.2.4 2xx Responses/UAC core establishes session with ACK";
netIn:SIPRes
[msg.status
requireRespo
cle after B timeout";
rIn:UserInput
g.cmd=="bye"]/
netIn:SIPRes
[msg.status
requireResp
ack(msg);
netIn:SIPRequest[msg.startLine.method == "BYE"]/
```

Loading the Model



Qtronic - Coverage Goals for SIPClient - Eclipse Platform

Project Explorer:

- SIPClient
  - DC 1
    - TTCNScripter
  - model
    - SIPClient.java
    - SIPClient.xmi
  - QtronicTypes.ttcn3
  - TestSuite.ttcn3

Coverage Editor: SIPClient

Testing Goals	DC 1	
Requirements	✓	0
13.2.2.4 2xx Responses	✓	0
UAC core establishes session with ACK	✓	⚠
15.1 Terminating a session	✓	0
17.1.1.2 INVITE timers	✓	0
17.1.2.2 Non-INVITE timers	✓	0
State Chart	✓	0
Conditional Branching	✓	0
Control Flow	✓	0

Trace:

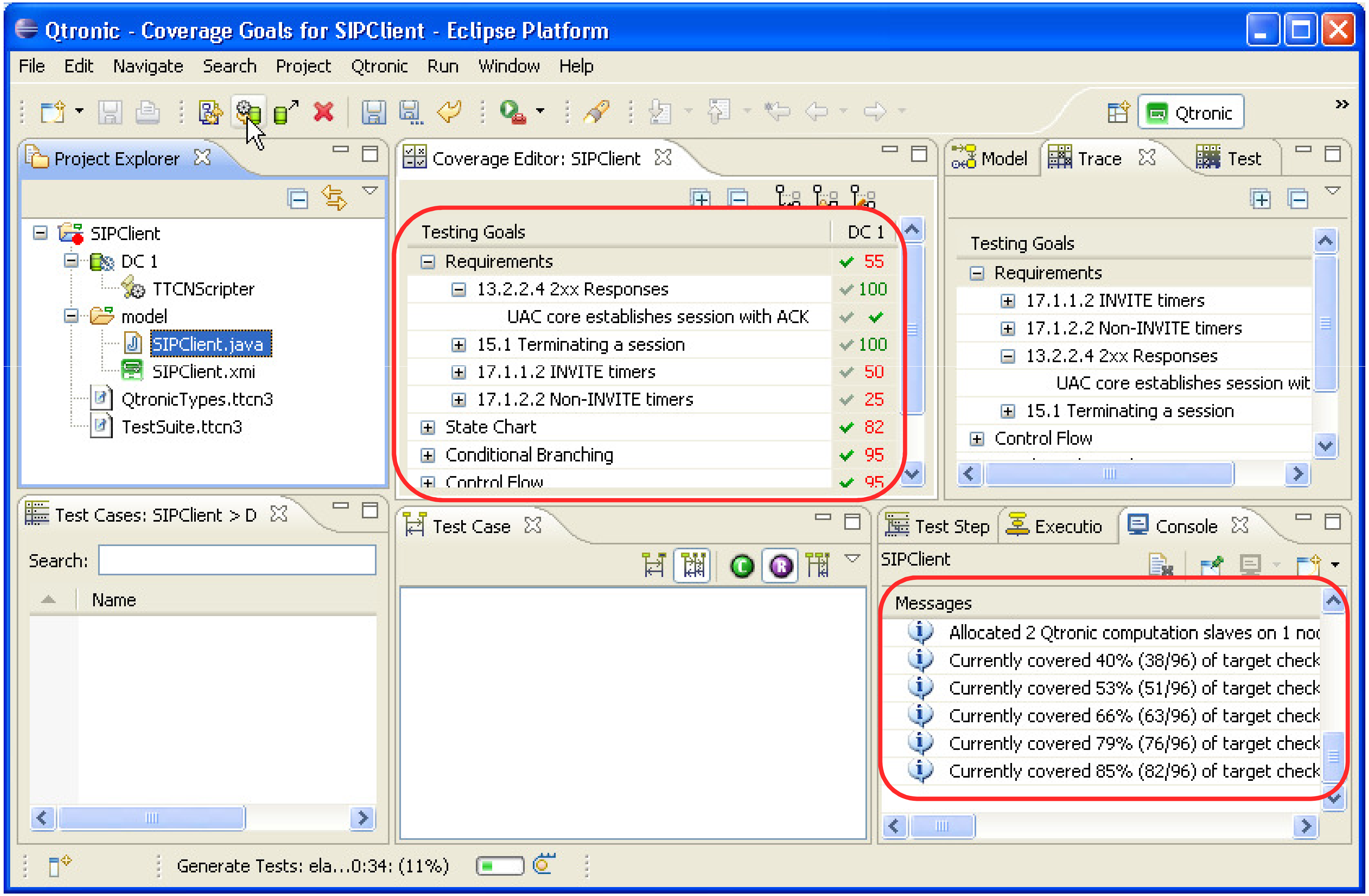
- Testing Goals
  - Requirements
    - 17.1.1.2 INVITE timers
    - 17.1.2.2 Non-INVITE timers
    - 13.2.2.4 2xx Responses
      - UAC core establishes session wit
    - 15.1 Terminating a session
  - Control Flow

Console: SIPClient

Messages
- Parsing the model...
- Type checking the model...
- Constructing IR graph...

Load Model: elapse...g 0:02: (35%)

Generating Tests from Models



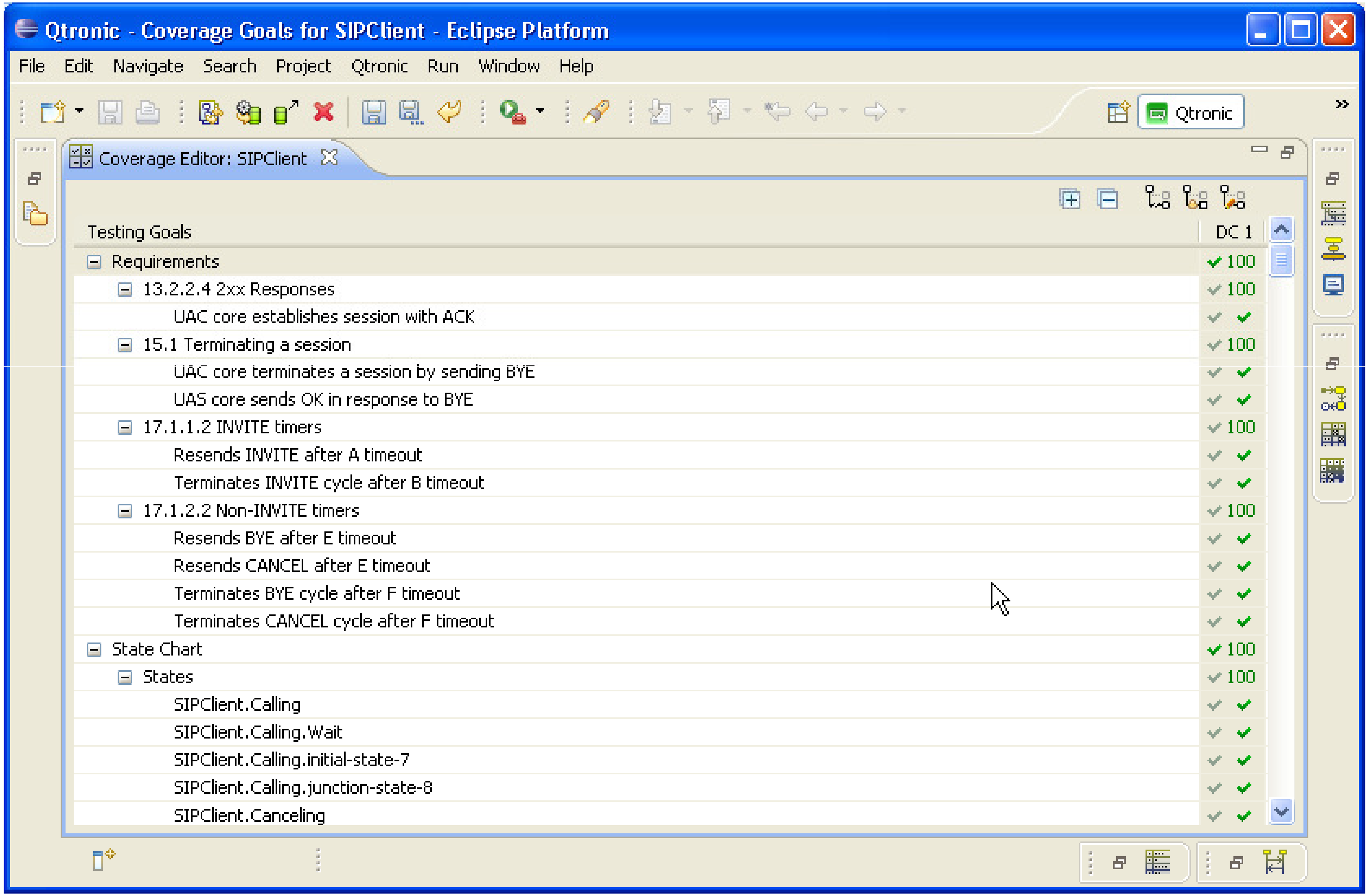
Testing Goals	DC 1
⊟ Requirements	✓ 55
⊟ 13.2.2.4 2xx Responses	✓ 100
UAC core establishes session with ACK	✓ ✓
⊞ 15.1 Terminating a session	✓ 100
⊞ 17.1.1.2 INVITE timers	✓ 50
⊞ 17.1.2.2 Non-INVITE timers	✓ 25
⊞ State Chart	✓ 82
⊞ Conditional Branching	✓ 95
⊞ Control Flow	✓ 95

Testing Goals
- ⊟ Requirements
  - ⊞ 17.1.1.2 INVITE timers
  - ⊞ 17.1.2.2 Non-INVITE timers
  - ⊟ 13.2.2.4 2xx Responses
    - UAC core establishes session wit
  - ⊞ 15.1 Terminating a session
- ⊞ Control Flow

Messages
- Allocated 2 Qtronic computation slaves on 1 noc
- Currently covered 40% (38/96) of target check
- Currently covered 53% (51/96) of target check
- Currently covered 66% (63/96) of target check
- Currently covered 79% (76/96) of target check
- Currently covered 85% (82/96) of target check

Generate Tests: ela...0:34: (11%)

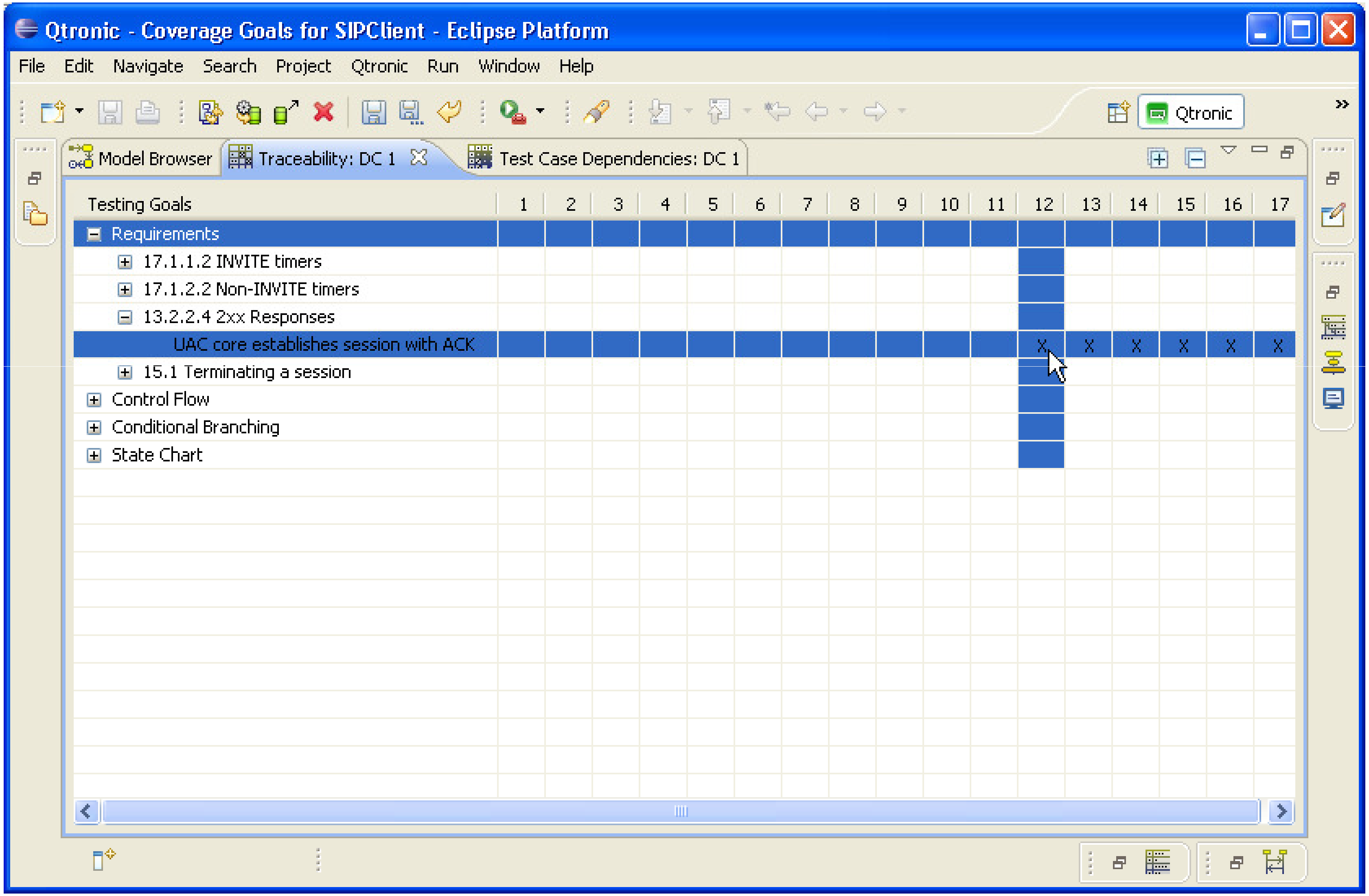
Results: Coverage Editor



Qtronic - Coverage Goals for SIPClient - Eclipse Platform
File Edit Navigate Search Project Qtronic Run Window Help
Qtronic
Coverage Editor: SIPClient

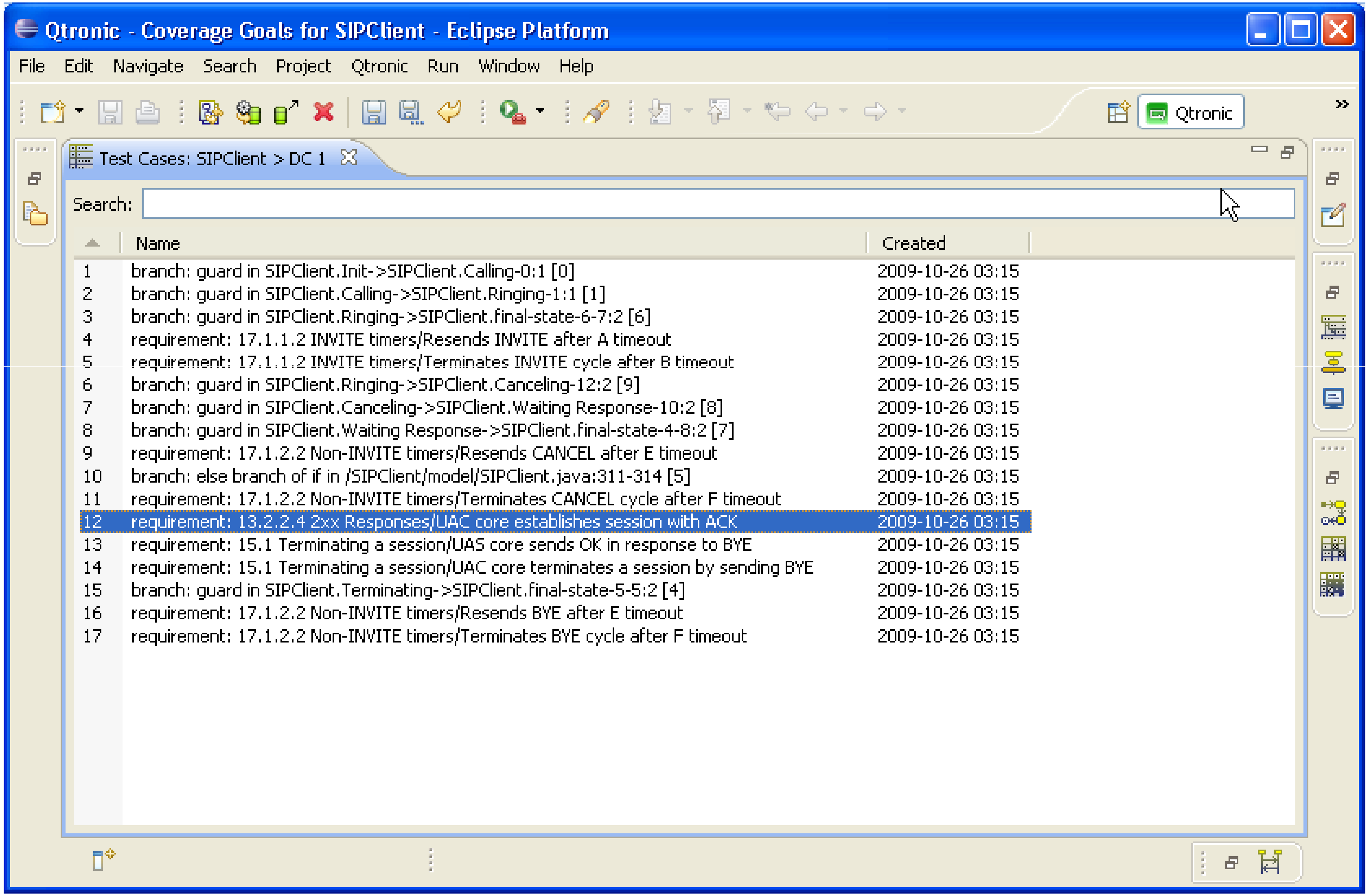
Testing Goals	DC 1
⊟ Requirements	✓ ✓ 100
⊟ 13.2.2.4 2xx Responses	✓ ✓ 100
UAC core establishes session with ACK	✓ ✓
⊟ 15.1 Terminating a session	✓ ✓ 100
UAC core terminates a session by sending BYE	✓ ✓
UAS core sends OK in response to BYE	✓ ✓
⊟ 17.1.1.2 INVITE timers	✓ ✓ 100
Resends INVITE after A timeout	✓ ✓
Terminates INVITE cycle after B timeout	✓ ✓
⊟ 17.1.2.2 Non-INVITE timers	✓ ✓ 100
Resends BYE after E timeout	✓ ✓
Resends CANCEL after E timeout	✓ ✓
Terminates BYE cycle after F timeout	✓ ✓
Terminates CANCEL cycle after F timeout	✓ ✓
⊟ State Chart	✓ ✓ 100
⊟ States	✓ ✓ 100
SIPClient.Calling	✓ ✓
SIPClient.Calling.Wait	✓ ✓
SIPClient.Calling.initial-state-7	✓ ✓
SIPClient.Calling.junction-state-8	✓ ✓
SIPClient.Canceling	✓ ✓

Results: Requirements Traceability Matrix


Qtronic - Coverage Goals for SIPClient - Eclipse Platform
File Edit Navigate Search Project Qtronic Run Window Help
Qtronic
Model Browser
Traceability: DC 1
Test Case Dependencies: DC 1

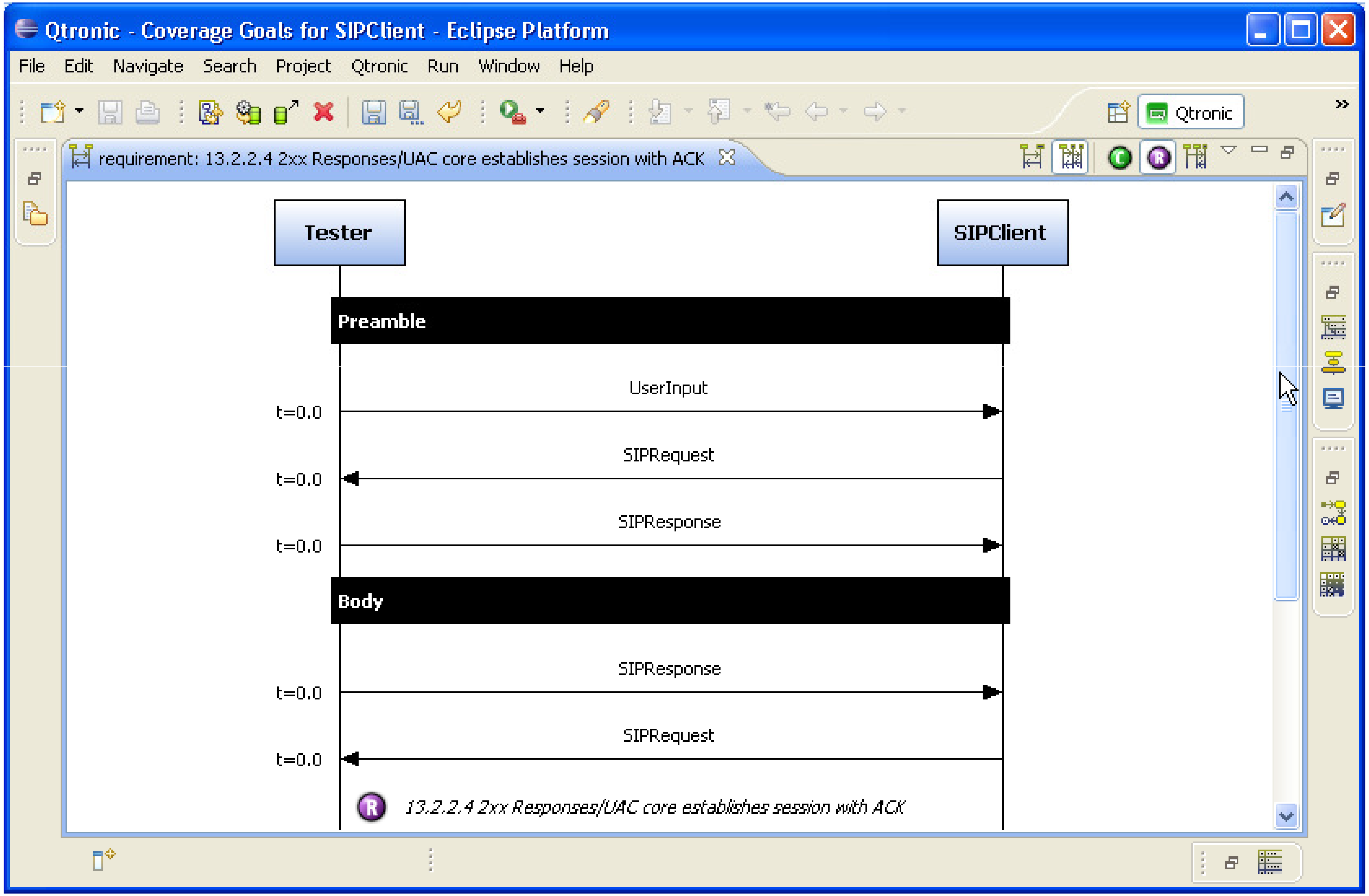
Testing Goals	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
⊟ Requirements																	
⊞ 17.1.1.2 INVITE timers																	
⊞ 17.1.2.2 Non-INVITE timers																	
⊟ 13.2.2.4 2xx Responses																	
UAC core establishes session with ACK												X	X	X	X	X	X
⊞ 15.1 Terminating a session																	
⊞ Control Flow																	
⊞ Conditional Branching																	
⊞ State Chart																	

Results: Test Case List


Qtronic - Coverage Goals for SIPClient - Eclipse Platform
File Edit Navigate Search Project Qtronic Run Window Help
Qtronic
Test Cases: SIPClient > DC 1
Search:

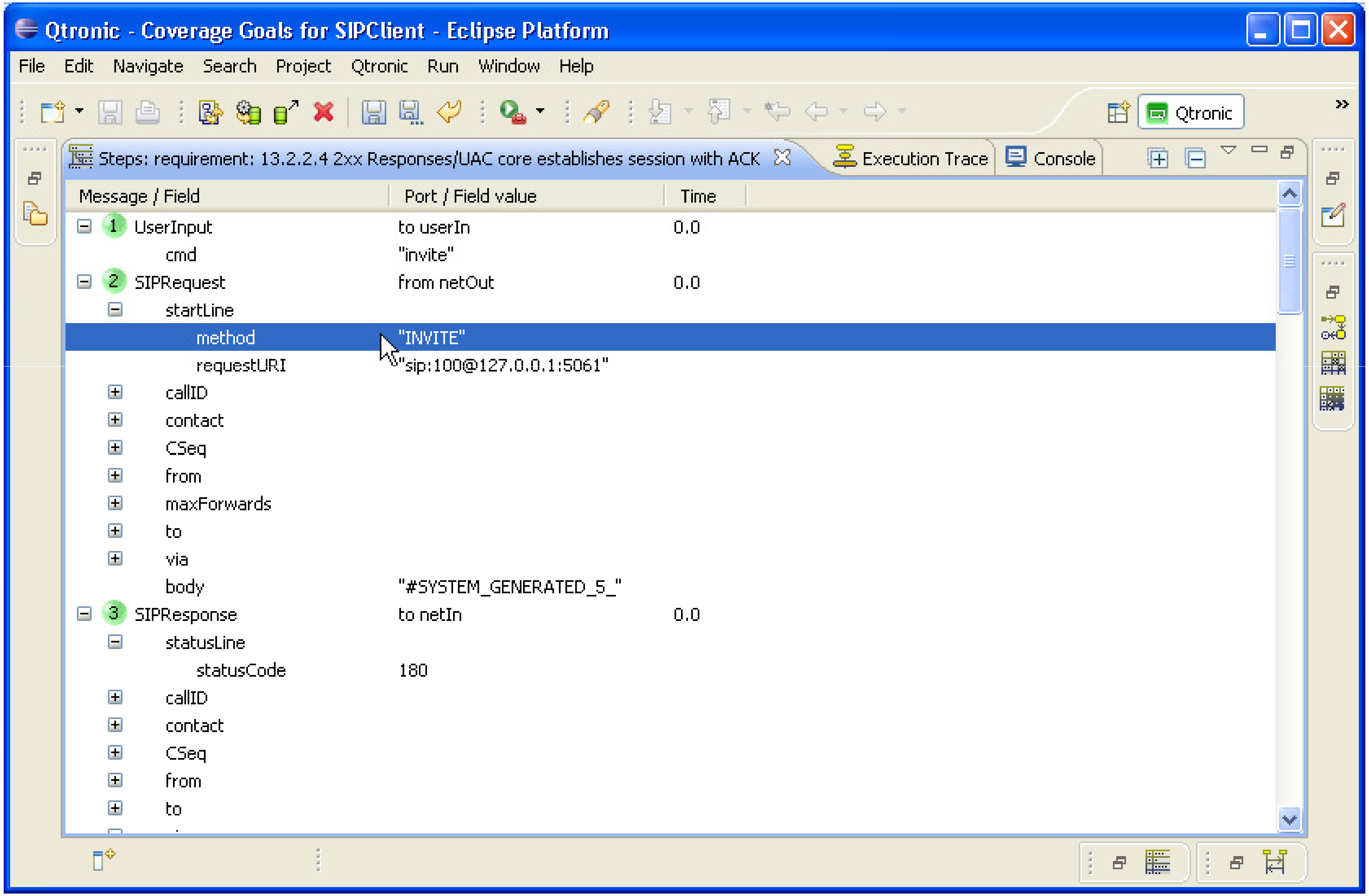
	Name	Created
1	branch: guard in SIPClient.Init->SIPClient.Calling-0:1 [0]	2009-10-26 03:15
2	branch: guard in SIPClient.Calling->SIPClient.Ringing-1:1 [1]	2009-10-26 03:15
3	branch: guard in SIPClient.Ringing->SIPClient.final-state-6-7:2 [6]	2009-10-26 03:15
4	requirement: 17.1.1.2 INVITE timers/Resends INVITE after A timeout	2009-10-26 03:15
5	requirement: 17.1.1.2 INVITE timers/Terminates INVITE cycle after B timeout	2009-10-26 03:15
6	branch: guard in SIPClient.Ringing->SIPClient.Canceling-12:2 [9]	2009-10-26 03:15
7	branch: guard in SIPClient.Canceling->SIPClient.Waiting Response-10:2 [8]	2009-10-26 03:15
8	branch: guard in SIPClient.Waiting Response->SIPClient.final-state-4-8:2 [7]	2009-10-26 03:15
9	requirement: 17.1.2.2 Non-INVITE timers/Resends CANCEL after E timeout	2009-10-26 03:15
10	branch: else branch of if in /SIPClient/model/SIPClient.java:311-314 [5]	2009-10-26 03:15
11	requirement: 17.1.2.2 Non-INVITE timers/Terminates CANCEL cycle after F timeout	2009-10-26 03:15
12	requirement: 13.2.2.4 2xx Responses/UAC core establishes session with ACK	2009-10-26 03:15
13	requirement: 15.1 Terminating a session/UAS core sends OK in response to BYE	2009-10-26 03:15
14	requirement: 15.1 Terminating a session/UAC core terminates a session by sending BYE	2009-10-26 03:15
15	branch: guard in SIPClient.Terminating->SIPClient.final-state-5-5:2 [4]	2009-10-26 03:15
16	requirement: 17.1.2.2 Non-INVITE timers/Resends BYE after E timeout	2009-10-26 03:15
17	requirement: 17.1.2.2 Non-INVITE timers/Terminates BYE cycle after F timeout	2009-10-26 03:15

Results: Abstract Test Case View



Qtronic - Coverage Goals for SIPClient - Eclipse Platform
File Edit Navigate Search Project Qtronic Run Window Help
Qtronic
requirement: 13.2.2.4 2xx Responses/UAC core establishes session with ACK
Tester
SIPClient
Preamble
UserInput
t=0.0
SIPRequest
t=0.0
SIPResponse
t=0.0
Body
SIPResponse
t=0.0
SIPRequest
t=0.0
R
13.2.2.4 2xx Responses/UAC core establishes session with ACK

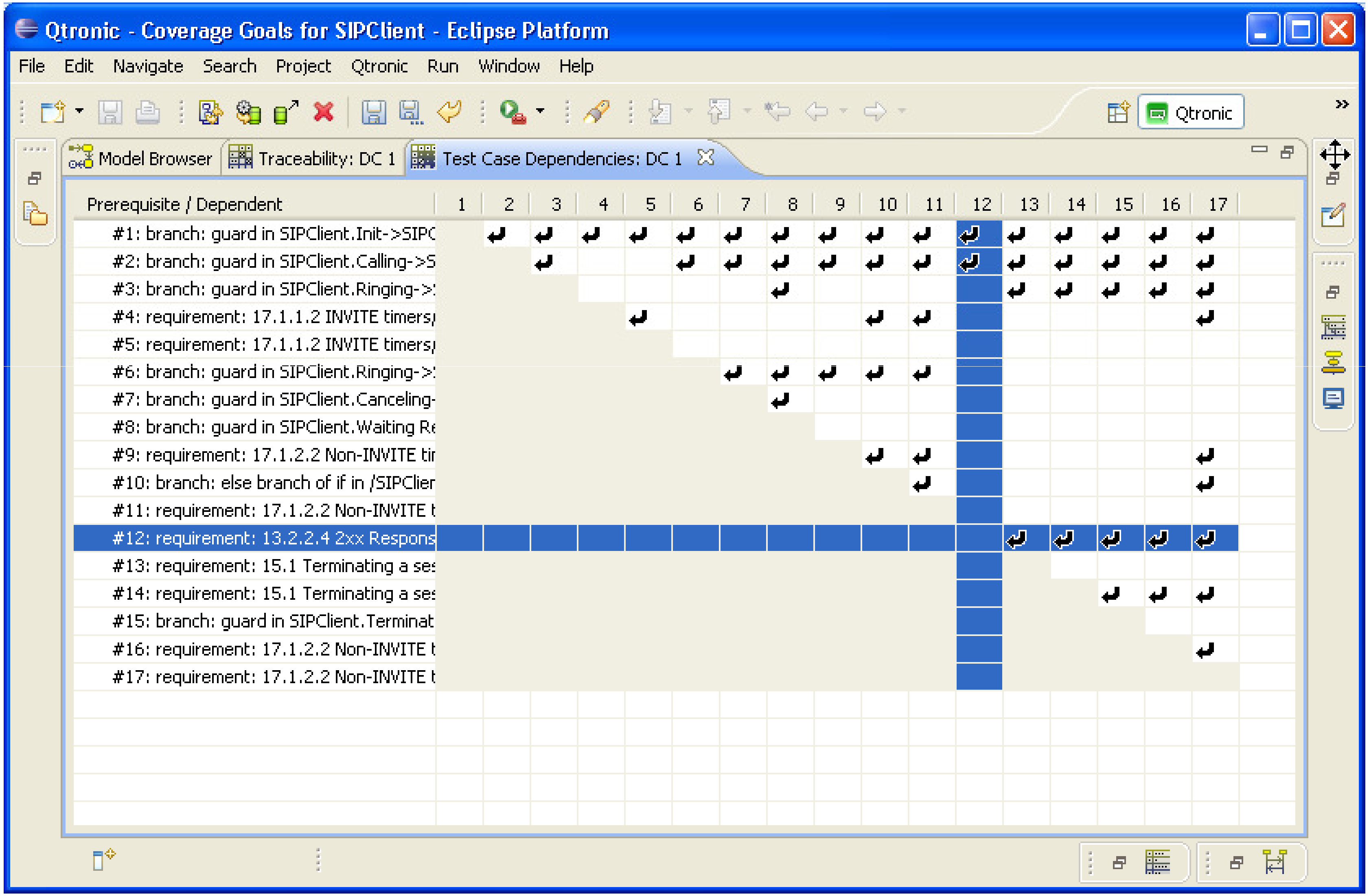
Results: Test Steps and Test Data



Qtronic - Coverage Goals for SIPClient - Eclipse Platform
File Edit Navigate Search Project Qtronic Run Window Help
Qtronic
Steps: requirement: 13.2.2.4 2xx Responses/UAC core establishes session with ACK
Execution Trace
Console

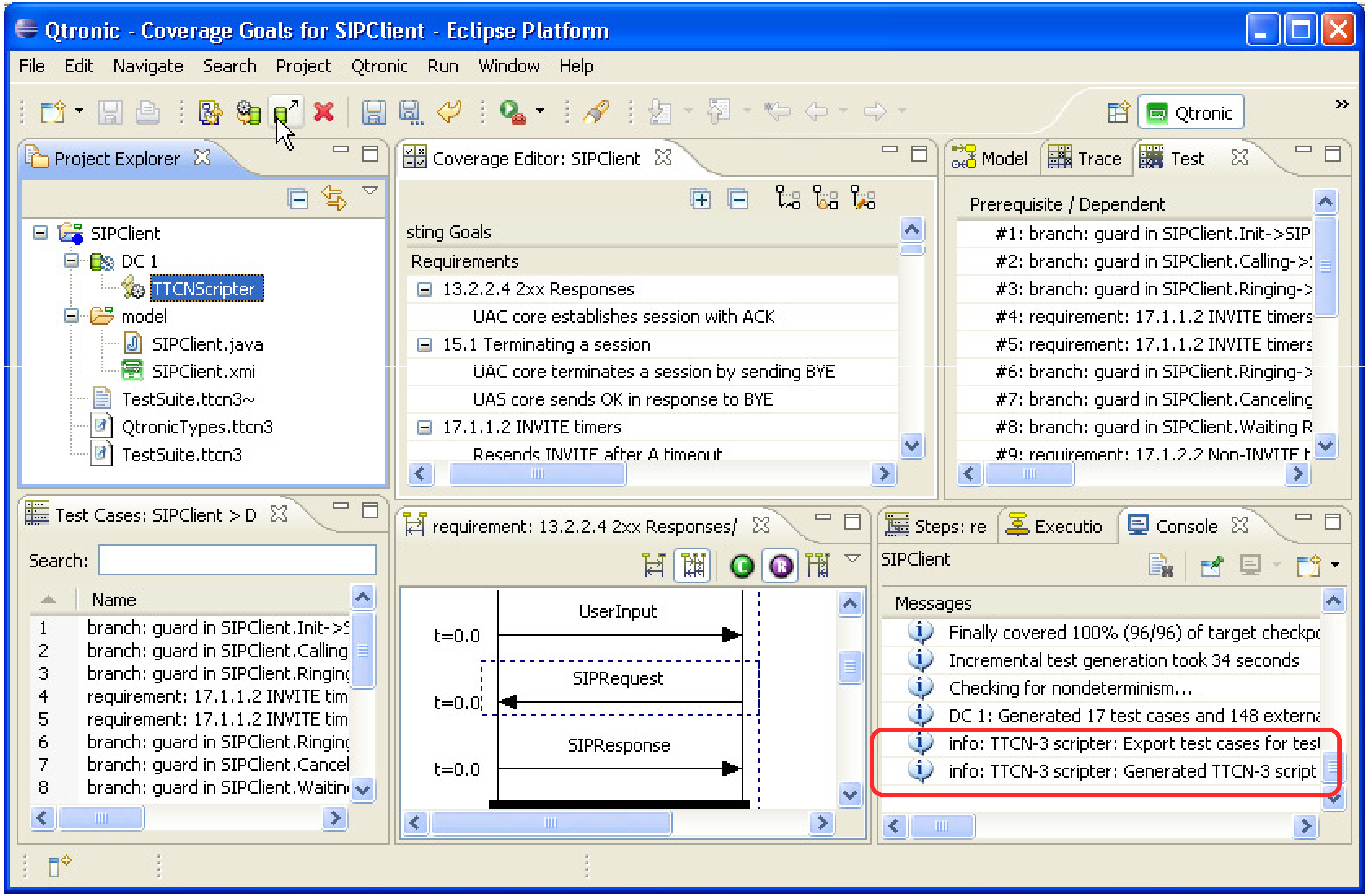
Message / Field	Port / Field value	Time
1 UserInput	to userIn	0.0
cmd	"invite"	
2 SIPRequest	from netOut	0.0
startLine		
method	"INVITE"	
requestURI	"sip:100@127.0.0.1:5061"	
callID		
contact		
CSeq		
from		
maxForwards		
to		
via		
body	"#SYSTEM_GENERATED_5_"	
3 SIPResponse	to netIn	0.0
statusLine		
statusCode	180	
callID		
contact		
CSeq		
from		
to		

Results: Test Case Dependency Matrix


Qtronic - Coverage Goals for SIPClient - Eclipse Platform
File Edit Navigate Search Project Qtronic Run Window Help
Qtronic
Model Browser
Traceability: DC 1
Test Case Dependencies: DC 1

Prerequisite / Dependent	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
#1: branch: guard in SIPClient.Init->SIPC		↵	↵	↵	↵	↵	↵	↵	↵	↵	↵	↵	↵	↵	↵	↵	↵
#2: branch: guard in SIPClient.Calling->S			↵			↵	↵	↵	↵	↵	↵	↵	↵	↵	↵	↵	↵
#3: branch: guard in SIPClient.Ringing->:								↵					↵	↵	↵	↵	↵
#4: requirement: 17.1.1.2 INVITE timers,					↵					↵	↵						↵
#5: requirement: 17.1.1.2 INVITE timers,																	
#6: branch: guard in SIPClient.Ringing->:							↵	↵	↵	↵	↵						
#7: branch: guard in SIPClient.Canceling-								↵									
#8: branch: guard in SIPClient.Waiting Re																	
#9: requirement: 17.1.2.2 Non-INVITE tir										↵	↵						↵
#10: branch: else branch of if in /SIPClier											↵						↵
#11: requirement: 17.1.2.2 Non-INVITE t																	
#12: requirement: 13.2.2.4 2xx Respons													↵	↵	↵	↵	↵
#13: requirement: 15.1 Terminating a ses																	
#14: requirement: 15.1 Terminating a ses															↵	↵	↵
#15: branch: guard in SIPClient.Terminat																	
#16: requirement: 17.1.2.2 Non-INVITE t																	↵
#17: requirement: 17.1.2.2 Non-INVITE t																	

Rendering the Tests as TTCN-3


Qtronic - Coverage Goals for SIPClient - Eclipse Platform
File
Edit
Navigate
Search
Project
Qtronic
Run
Window
Help
Qtronic
Project Explorer
Coverage Editor: SIPClient
Model
Trace
Test
Test Cases: SIPClient > D
requirement: 13.2.2.4 2xx Responses/
Steps: re
Executio
Console

Project Explorer:

- SIPClient
  - DC 1
    - TTCNScripter
  - model
    - SIPClient.java
    - SIPClient.xmi
  - TestSuite.ttcn3~
  - QtronicTypes.ttcn3
  - TestSuite.ttcn3

Coverage Editor: SIPClient

sting Goals

Requirements

- 13.2.2.4 2xx Responses
  - UAC core establishes session with ACK
- 15.1 Terminating a session
  - UAC core terminates a session by sending BYE
  - UAS core sends OK in response to BYE
- 17.1.1.2 INVITE timers
  - Resends INVITE after A timeout

Prerequisite / Dependent

#1: branch: guard in SIPClient.Init->SIP
#2: branch: guard in SIPClient.Calling->:
#3: branch: guard in SIPClient.Ringing->
#4: requirement: 17.1.1.2 INVITE timers
#5: requirement: 17.1.1.2 INVITE timers
#6: branch: guard in SIPClient.Ringing->
#7: branch: guard in SIPClient.Canceling
#8: branch: guard in SIPClient.Waiting R
#9: requirement: 17.1.2.2 Non-INVITE t

Test Cases: SIPClient > D

Search:

	Name
1	branch: guard in SIPClient.Init->S
2	branch: guard in SIPClient.Calling
3	branch: guard in SIPClient.Ringing
4	requirement: 17.1.1.2 INVITE tim
5	requirement: 17.1.1.2 INVITE tim
6	branch: guard in SIPClient.Ringing
7	branch: guard in SIPClient.Cancel
8	branch: guard in SIPClient.Waitin

requirement: 13.2.2.4 2xx Responses/

t=0.0 UserInput

t=0.0 SIPRequest

t=0.0 SIPResponse

Console

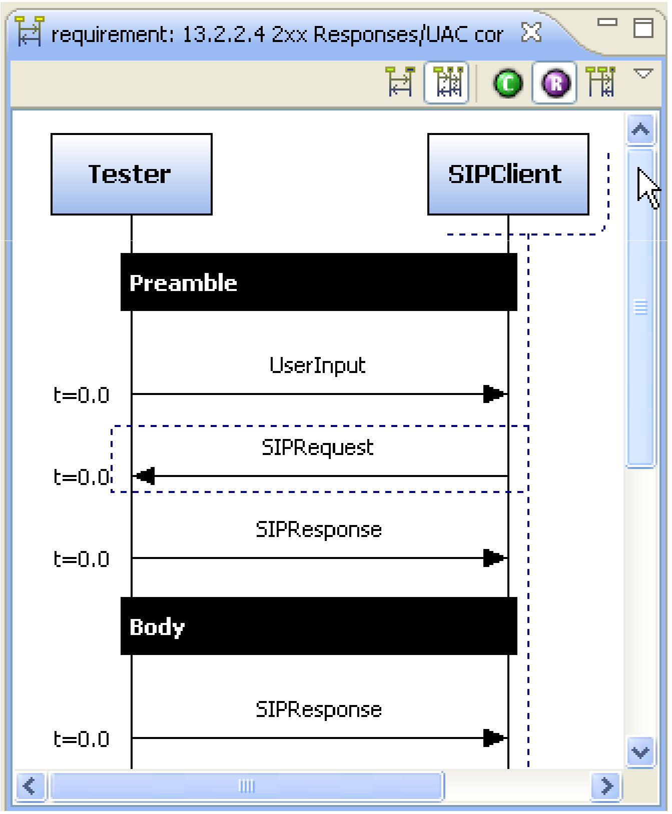
SIPClient

Messages

- Finally covered 100% (96/96) of target checkp
- Incremental test generation took 34 seconds
- Checking for nondeterminism...
- DC 1: Generated 17 test cases and 148 extern
- info: TTCN-3 scripter: Export test cases for tes
- info: TTCN-3 scripter: Generated TTCN-3 script

TTCN-3 Test Case

CQ Designer Test Case View

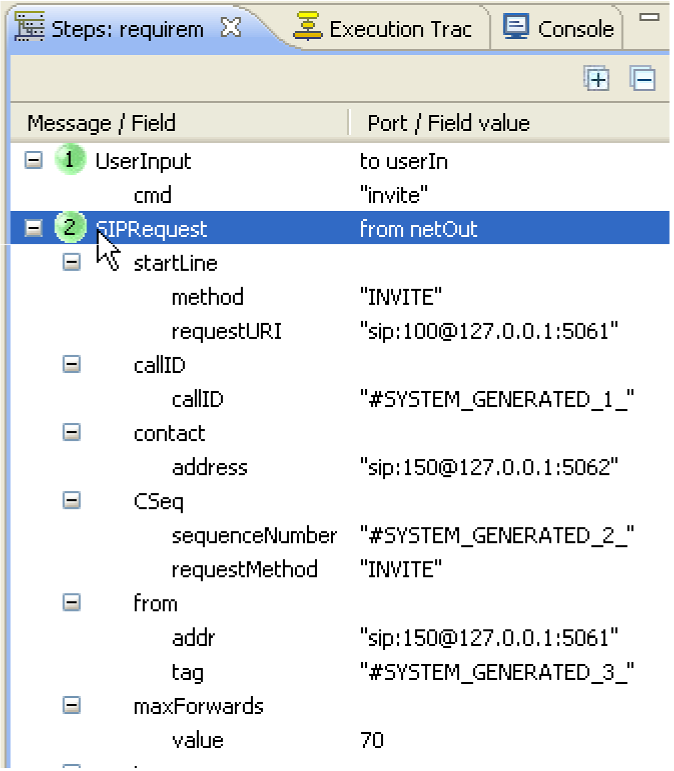


Rendered TTCN-3 Test Case

```
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_expecte
  t_cq_timer.stop; v_lastTimeout := 0.0;
  f_send_SIPResponse_to_netIn(m_SIPResponse9
  t_cq_timer.start((0.0 – v_last_timeout) +
  f_receive_UserOutput_from_userOut(m_expect
  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_expecte
  t_cq_timer.stop; v_lastTimeout := 0.0;
  log(“requirement: 13.2.2.4 2xx Response/UA
  ...
}
```

TTCN-3 Test Data

Message Data in CQ Designer



Message / Field	Port / Field value
1 UserInput	to userIn
cmd	"invite"
2 SIPRequest	from netOut
startLine	
method	"INVITE"
requestURI	"sip:100@127.0.0.1:5061"
callID	
callID	"#SYSTEM_GENERATED_1_"
contact	
address	"sip:150@127.0.0.1:5062"
CSeq	
sequenceNumber	"#SYSTEM_GENERATED_2_"
requestMethod	"INVITE"
from	
addr	"sip:150@127.0.0.1:5061"
tag	"#SYSTEM_GENERATED_3_"
maxForwards	
value	70

Rendered TTCN-3 message

```
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
          requestMethod := "INVITE"
     },
     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

Q&A

Contact Information

Jani Koivulainen

Director, Customer Success

jani.koivulainen@conformiq.com

+358408654351

http://www.conformiq.com