

**mVerify**<sup>TM</sup>

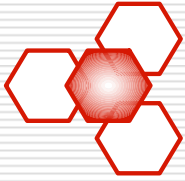
*A Million Users in a Box* ®

Test  
^ **Objects -- They Just Work**

September 8, 2006

Google Test Automation Conference

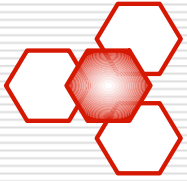
[www.mverify.com](http://www.mverify.com)



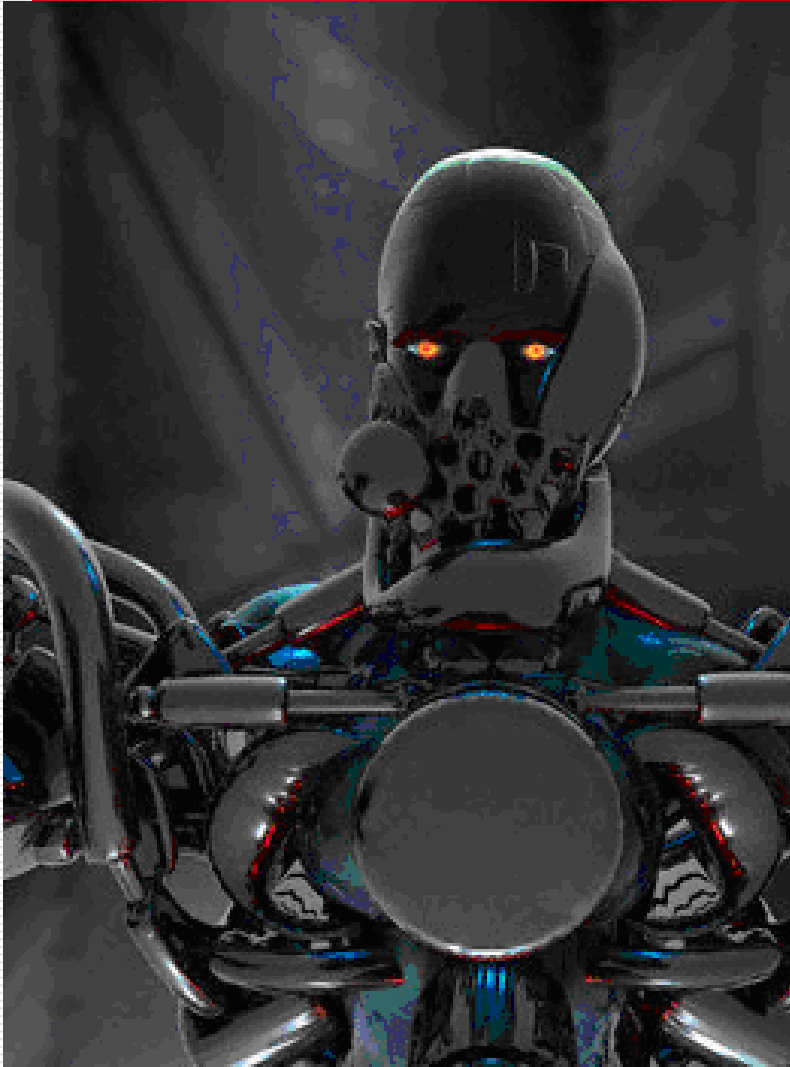
# Overview

---

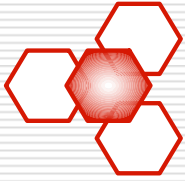
- Motivation
- MTS Goals
- TTCN Influences
- X-Unit Influences
- MTS::TestObject
- Demo
- Q & A



# The Mobile Testing Nightmare



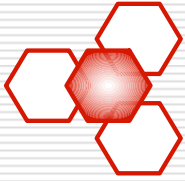
- ❑ Intense, high-stakes race to market
- ❑ Configurations (platforms x devices x airlinks) increase exponentially
- ❑ More testing necessary for competitive quality, reliability, performance
- ❑ Ad hoc manual testing is slow, costly, ineffective



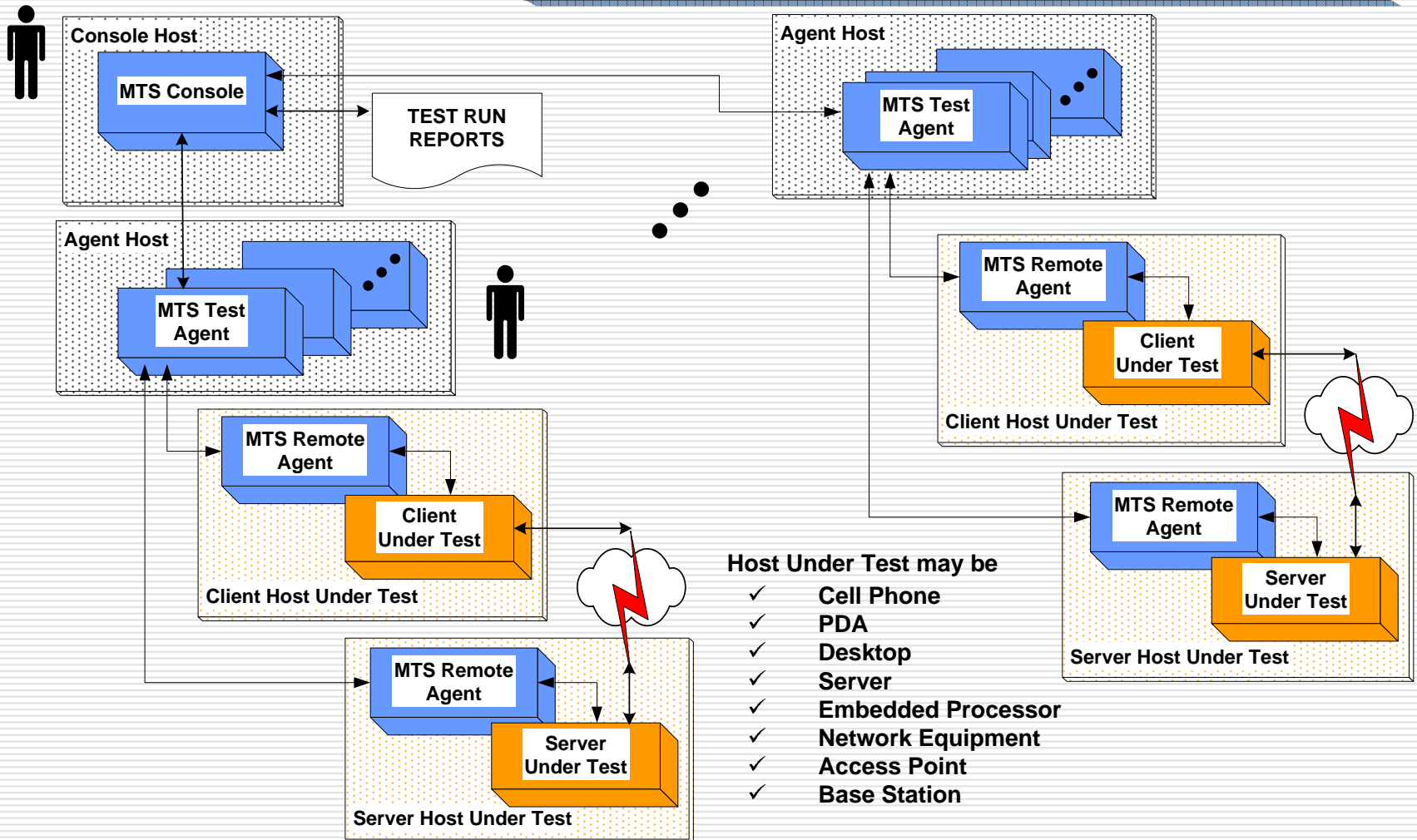
# MTS Background

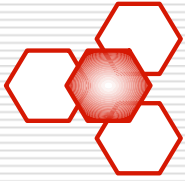
---

- Advanced model-based, mobile-centric testing
  - Necessary for 21<sup>st</sup> century technology
  - It's time for a change
  - Model-based, mobile-centric, reliability-maximizing
- Advanced test strategy useless without
  - End to end and embedded observation & control
  - Scalable, Robust
  - Distributed
  - Broadly usable



# MTS: Any App, Any Platform

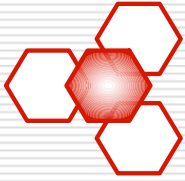




# Design Goals

---

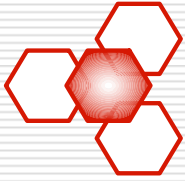
- Generate Test Objects from models
- Generate models from IUT
- Logical/Physical separation
- Platform agnostic/robust
- Channel agnostic/robust
- Minimize IUT footprint
- Distributed control
- Works out of the box/Agile
- Intuitive interaction
- Minimize brittleness
- Composable



# About TTCN-3

---

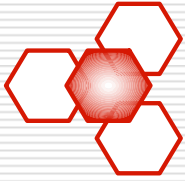
- Testing and Test Control Notation, international standard
- Abstract specification
- Structured/Procedural paradigm
- Proven support for complex scenarios
- Proven support for concurrency
- ASN.1 data (packed, maps to XML)
- Routine use in very high-reliability applications
  - Rigorous conformance testing of protocol implementations, from single test spec



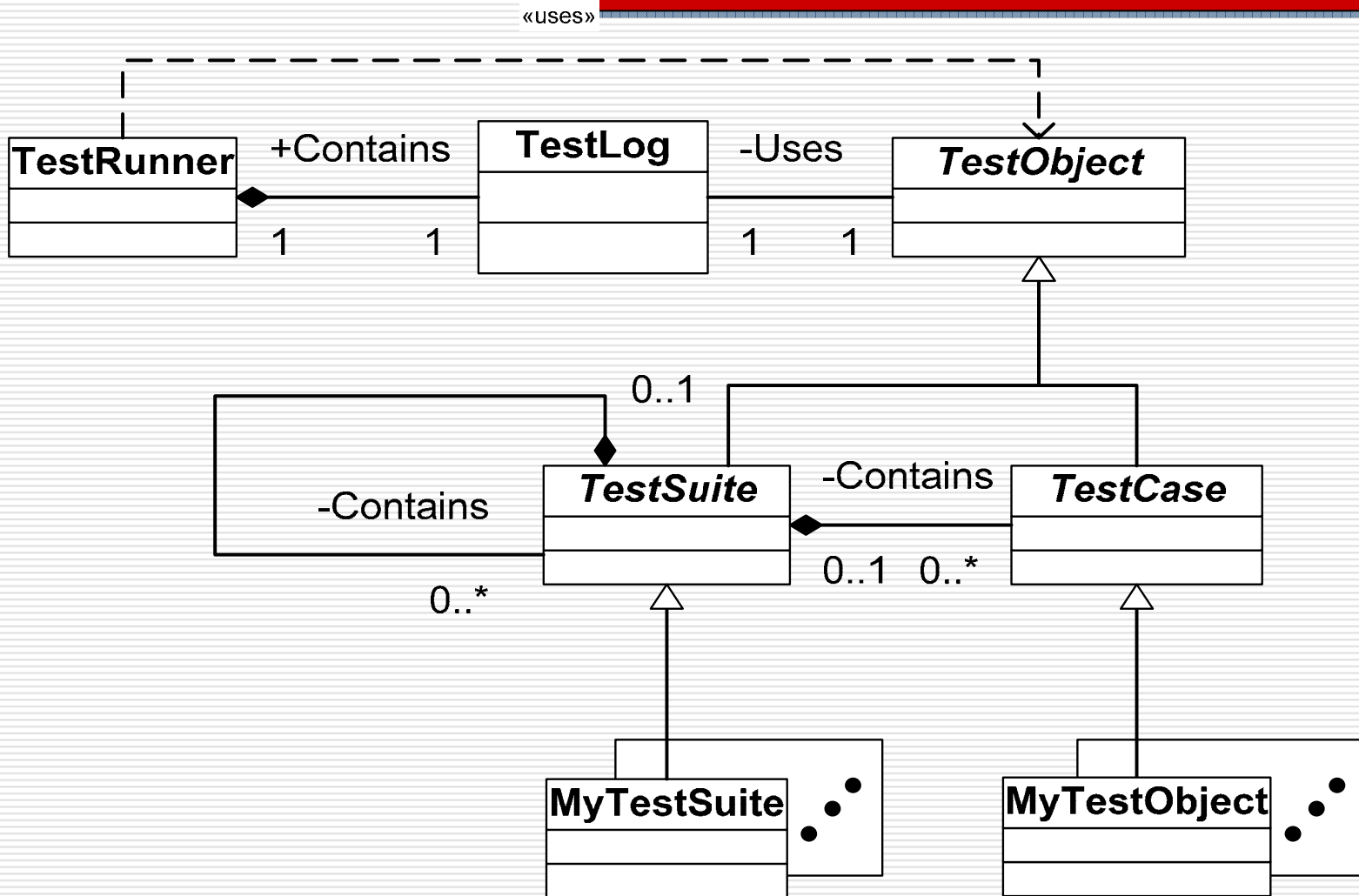
# About TTCN-3

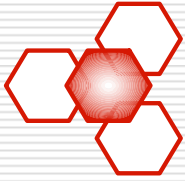
---

- Point of Observation and Control
  - Abstract Interface Specification
  - Each user provides physical binding for logical interface definition
  - *Same test suites can be used on many implementations*
- Model-based Testing
  - Generate TTCN from MSCs, FSMs, etc.
  - Compile to implementation language (C++, Java)
  - Compile & run implementation



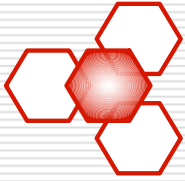
# Incremental Testing Framework





# TTCN and ITF -- Complementary

<b><i>MTS Design Goal</i></b>	<b><i>TTCN</i></b>	<b><i>ITF</i></b>
Automate test object generation	Yes	NA
Logical/Physical separation	Yes	No
Platform agnostic/robust	Yes	No
Channel agnostic/robust	Yes	NA
Distributed control	Yes	No
Minimize IUT footprint	NA	No
Works out of the box/Agile	No	Yes
Intuitive interaction	No	Yes
Composable	No	Yes
Minimize brittleness	No	No



# TTCN and ITF -- Limitations

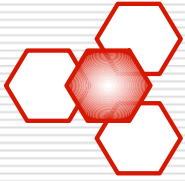
---

## □ TTCN

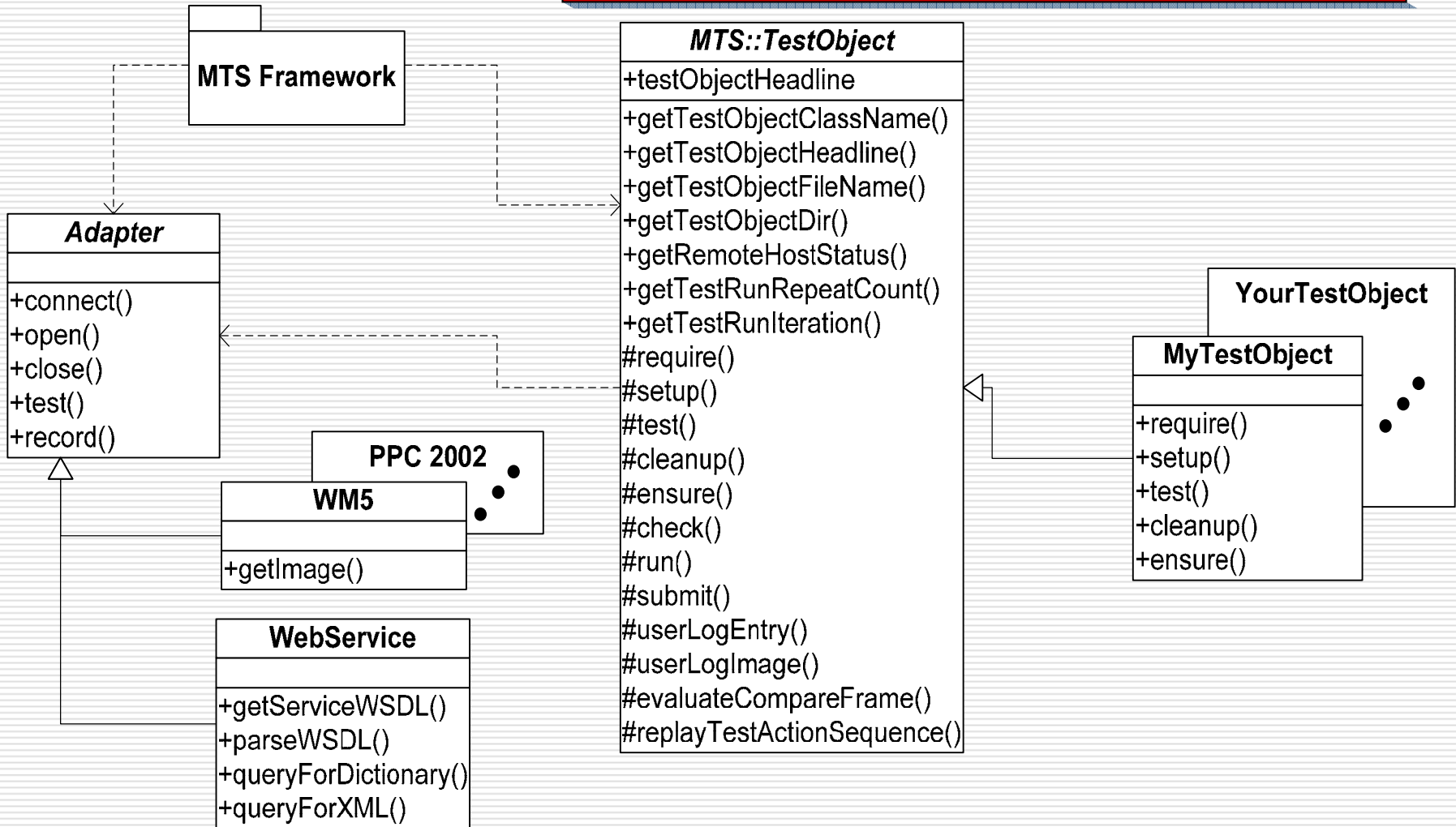
- Not executable – must be compiled
- L2P implementation out of scope
- Modular, but not composable

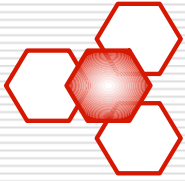
## □ ITF

- TestSuite limited to static object list
- Not control-composable
- Tight coupling with IUT
- Greedy
- Language-centric
- Developer-centric



# MTS::TestObject

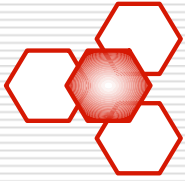




# MTS::TestObject

---

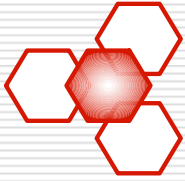
- ❑ Adapter: built-in remote host proxy; connection management, logging, exception handling
- ❑ Control-composable `TestObject::run <testobject>`
- ❑ Pre-conditions `TestObject::require`
- ❑ Post-conditions `TestObject::ensure`
- ❑ Built-in data-driven test from literal, variable, or function  
`TestObject::submit <data_provider>`
- ❑ Pass/fail any Tcl expression: `TestObject::check <expr>`
- ❑ XML test run log
- ❑ Incr Tcl, any Tcl package, library extension, or procedure
- ❑ Support stories, suites, cases, procedures ...



# Demo

---

- Display output resulting from `MTS::TestObject` capabilities
  - Adapter selection
  - One-click Repeat
  - Smart Progress Bar
- Much more ...



# Demo: One Test Object, One Time

- ❑ MTS::TestObject has 5 steps: require, setup, test, cleanup, ensure
- ❑ Normal completion all steps, Smart Progress Bar shows all **green**
- ❑ Same as ITF test execution

The screenshot shows the MTS Test Agent 3936 Ready application window. The window title is "MTS Test Agent 3936 Ready" and it has a menu bar with "Agent", "Remote Host", "Test Objects", "Test Results", and "Help".

**Remote Host**

- Headline: [Empty]
- Device Type: Not active
- Id: [Empty]
- Nickname: [Empty]
- Refresh Rate (fps): 1
- Connection: Disconnected
- Actions: 0
- Data: Not active

**Current Test Object**

- Headline: Passes all, skip after 42 repetitions
- File: ..\MTS.130\Repository\TestAssets\GLTAC\Green.mvto
- Repeat:  Repeat
- Objects: 0
- Actions: 0
- Checks: 0

**Test Run Results**

- Progress Bar: 100% (Green)
- Steps: 6
- Passed: 2
- Failed: 0
- Exceptions: 0
- Run Time: 0:00:00

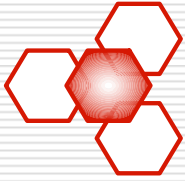
**Test Run Log**

```
2006/09/11 08:48:48 END Green::cleanup step
2006/09/11 08:48:48 ENSURE
2006/09/11 08:48:48 NOTE Green::ensure postcondition met
2006/09/11 08:48:48 END Green::ensure step
2006/09/11 08:48:48 END Test object Passes all, skip after 42 repetitions
```

**RunTime Output Monitor**

```
2006/09/11 08:48:38 info tid000000E4 Test object ..\MTS.130\Repository\TestAssets\GLTAC\Green.mvto load
2006/09/11 08:48:48 info tid000000E4 Running test object Green
2006/09/11 08:48:49 notice tid000000E4 Test run completed. Ready for next action
```

Test run completed. Ready for next action



# Demo: Stepwise Results

- What happens when a step has an error or failure?
- Smart Progress Bar shows yellow or red at relative progress increment

The screenshot shows the MTS Test Agent 3936 Ready interface. The window title is "MTS Test Agent 3936 Ready". The menu bar includes "Agent", "Remote Host", "Test Objects", "Test Results", and "Help".

**Remote Host**

- Headline: [Empty]
- Device Type: Not active
- Id: [Empty]
- Nickname: [Empty]
- Refresh Rate (fps): 1
- Connection: Disconnected
- Actions: 0
- Data: Not active

**Current Test Object**

- Headline: Pass, Error, Fail once
- File: ..\Repository\TestAssets\GLTAC\GreenYellowRed.mvto
- Objects: 0
- Actions: 0
- Checks: 0
- Repeat:  Repeat
- Count: 2

**Test Run Results**

- Progress bar: 100% (Green, Yellow, Red)
- Steps: 6
- Passed: 1
- Failed: 1
- Exceptions: 1
- Run Time: 0:00:00

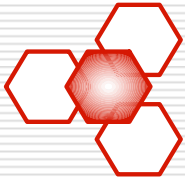
**Test Run Log**

```
2006/09/11 08:49:41 END GreenYellowRed::cleanup step
2006/09/11 08:49:41 ENSURE GreenYellowRed::ensure postcondition met
2006/09/11 08:49:41 NOTE GreenYellowRed::ensure step
2006/09/11 08:49:41 END GreenYellowRed::ensure step
2006/09/11 08:49:41 END Test object Pass, Error, Fail once
```

**RunTime Output Monitor**

```
2006/09/11 08:49:40 info tid000000E4 Test object ..\Repository\TestAssets\GLTAC\GreenYellowRed.mvto loa
2006/09/11 08:49:41 info tid000000E4 Running test object GreenYellowRed
2006/09/11 08:49:41 notice tid000000E4 Test run completed. Ready for next action
```

Test run completed. Ready for next action



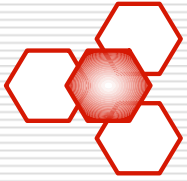
# Demo: Robust Control

- What happens when a TestObject runs under robust control?
- Smart Progress Bar shows yellow or red at relative progress increments, for each iteration

The screenshot shows the MTS Test Agent 3936 Ready application window. The window has a menu bar with 'Agent', 'Remote Host', 'Test Objects', 'Test Results', and 'Help'. The main content is divided into several sections:

- Remote Host:** Contains icons for various actions (connect, disconnect, refresh, etc.). Fields include 'Headline', 'Device Type' (Not active), 'Id', 'Nickname', 'Refresh Rate (fps)' (1), 'Connection' (Disconnected), 'Actions' (0), and 'Data' (Not active).
- Current Test Object:** Contains icons for test object actions. Fields include 'Headline' (Pass, Error, Fail once), 'File' (..\Repository\TestAssets\GLTAC\GreenYellowRed.mvto), 'Objects' (0), 'Actions' (0), 'Checks' (0), and a 'Repeat' checkbox with a value of 34.
- Test Run Results:** Features a progress bar showing 100.0% completion. Below the bar, it displays 'Steps' (204), 'Passed' (34), 'Failed' (34), 'Exceptions' (34), and 'Run Time' (0:00:03).
- Log Output:** A text area showing test execution logs, including 'END GreenYellowRed:cleanup step', 'ENSURE GreenYellowRed:ensure postcondition met', and 'Test object Pass, Error, Fail once'.
- RunTime Output Monitor:** A text area showing runtime output, including 'tid000000E4 Running repetition 33 of test object GreenYellowRed, 1 repetition', 'tid000000E4 Running repetition 34 of test object GreenYellowRed, 0 repetitions', and 'tid000000E4 Test run completed. Ready for next action'.

At the bottom of the window, a status bar reads 'Test run completed. Ready for next action'.



# Notes

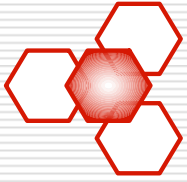
---

- NeXTSTEP demo, Steve Jobs says “It just works -- they’re objects”  
<http://rixstep.com/1/1/20060814,00.shtml>

- Mobile Testing Nightmare Poster

- Questions: [Bob\\_Binder@mverify.com](mailto:Bob_Binder@mverify.com)





Q & A