Command and Control Automated Testing (C2AT) Capabilities — Contact List

Visit our website (*www.eglin.af.mil/test_and_analysis_division*) to…

- See our current DT&E and OT&E capabilities,
- Request current product information,
- Update your mailing address,
- Submit a software change request, questions, comments, or suggestions, or
- Request replacement of missing or out-of-date items (disks, manuals, etc.).

You may also contact us by telephone, FAX, e-mail, or mail.

Test and Analysis Division Customer Service

Phone:	(850) 882-8470 DSN 872-8470
FAX:	(850) 882-5924 DSN 872-5924
E-mail:	eglin.testandanalysisdivision@eglin.af.mil
Mail:	Freeman Computer Sciences Center Attn: Test and Analysis Division Customer Service 201 W Eglin Blvd Suite 228 Eglin AFB FL 32542-6829

Command and Control Automated Testing (C2AT) Capabilities

Abstract

Command is the legal authority to issue orders. Control is the management of assets to accomplish mission objectives. Automated command and control systems assist the commander and the commander's staff in issuing orders and controlling assets. The Theater Battle Management Core System (TBMCS) is the Command and Control (C2) automated system used to assist commanders and their staffs (planners, operations, special operations, intelligence, and logistics) in managing all aspects of the air war.

The testing of TBMCS and other C2 systems is vital. We must ensure that automated C2 systems work properly to help ensure effective command and control. Command and Control Automated Testing (C2AT) ensures that the C2 software operated properly and performs well under full system load. If there are problems with the C2 automated system, the problems are identified and corrected with computer precision. Once the C2 system is corrected, the entire test can be re-run, to ensure that a correction to one part of the C2 system does not cause a problem in another part of the C2 system.

C2AT provides Automated Developmental Testing (ADT), Automated Operational Testing (AOT) and Automated Regression Testing (ART) in the Microsoft Windows and Sun Solaris, Linux, and HP-UX operating systems environment. Automated Test Management (ATM) testing tools are used to chronicle testing requirements, document test events, and execute automated test scripts. The description section of the automated test plan describes each test event. Test steps are developed to provide a detailed description of how each test event is performed. Skilled users perform the test events and their actions are recorded and saved as ADT test scripts. Data Base queries and/or user screen integrity checks are inserted into the ADT test scripts to provide more detailed test scripts for AOT/ART testing. The Automated Test Management software executes the automated test plan on a minimal hardware suite without human interaction. After the automated test is completed, an automated test report is generated into a Microsoft Word document. Automated Performance (load) Testing (APT) can also be accomplished on a small suite of computers simulating the load of hundreds of simultaneous users. Monitoring database servers, web servers, and computer networks using these automated test tools can give a complete picture of the entire computer system and confidence about its readiness for operational testing. Automated 24/7 System Monitoring (A24/7SM) can be accomplished using the C2AT tools. When discrepancies are found, automated test scripts can be re-run exactly how they were performed earlier to help further identify the cause of the problem(s). Once discrepancies are corrected, an ART can be run in a few hours to ensure that corrections did not cause other problems.

Traditionally, developmental and operational test events are often combined. Military and civilian personnel are sent to the test site, trained for one to two weeks, then asked to perform rote test scripts in a certain order and write down anything that they think is not correct. If discrepancies are identified, they are corrected and the software is assumed to be ready for fielding. During traditional test events, if response time seems slow to the test users, testers speculate about network performance, server performance, or a particular module of code. Often the real problem is not identified. Automated tools allow testers to identify problems without guesswork, including network delay problems, server memory

allocation and paging problems, CPU problems, database problems, web server problems, and problems that may occur under maximum load. Best of all, tests can be re-run at any time (even off duty hours), accurately, without sending personnel on temporary duty. Another benefit of using automated testing tools, when the next version of the software is ready for testing, the automated scripts can be re-run (often with little or no modification). Well-trained personnel cannot identify many performance, database, network, and programming issues with computer accuracy. Once accurate problem identification occurs, problem resolution can occur. This automated process greatly reduces development time and helps improve the fielding of applications.

C2AT Environment

C2AT uses Mercury Interactive software products to perform developmental, operational, regression, performance, and system monitoring testing (see Figure 1). Mercury Interactive products work in the Microsoft Windows, HP-UX, Sun Solaris, UNIX, and LINUX operating systems environments. Mercury products are fully integrated within the family of products and are Open Test Architecture (OTA) compliant. Programs written in all major languages, using Oracle, Sybase or Microsoft Access, or Microsoft SQL Server databases, and all the major web servers are compatible with Mercury Interactive software products. C2AT test tools include TestDirector (Microsoft Windows environment), WinRunner (Microsoft Windows environment), QuickTest Pro (Web Testing environment) XRunner (UNIX, HP-UX, and Solaris environments), LoadRunner (UNIX, Microsoft Windows, and LINUX environments) and Topaz (UNIX, Microsoft Windows, and LINUX environments). TestDirector, LoadRunner, and Topaz require one additional computer each to be added to your C2 network. WinRunner, QuickTest Pro and XRunner fit on actual C2 workstations. When possible it is better to put WinRunner, QuickTest Pro, and XRunner on a separate hard drive on a typical application workstation.

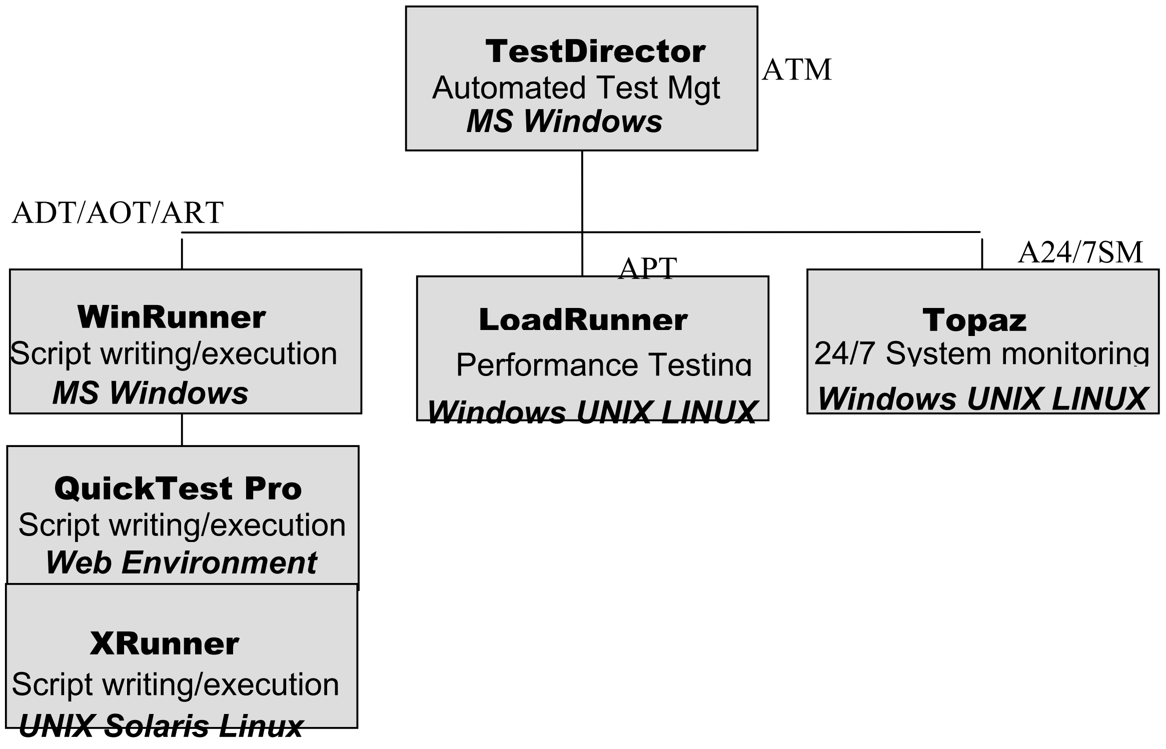


Figure 1—C2AT Schematic

Automated Developmental Testing Process

The Automated Developmental Testing process using automated tools is outlined in Figure 2. This process is test requirement driven. Test requirements are studied, then listed in TestDirector software. After listing requirements, the test plan is written in and/or copied to TestDirector software. Test scripts are developed based on client workstations (running WinRunner, QuickTest Pro, and/or XRunner). Test Scripts are written automatically in Test Script Language by recording user input. Integrity and database quality control checks can be added to WinRunner and/or XRunner test scripts. Then test scripts are imported into TestDirector. Test scripts are organized in the Test Lab module of TestDirector. Test Script execution is initiated and controlled by TestDirector. Test scripts run on client workstations (WinRunner, QuickTest Pro and/or XRunner). TestDirector software compiles the results. Test defects are detailed in the Defects module of TestDirector. Once defects are corrected, the entire test can be re-run, ensuring that a fix in one area does not cause a problem in another area of the automated system. Once all defects are eliminated, the developmental test is completed and archived for future releases of the Command and Control software. The diagram below demonstrates the developmental process.

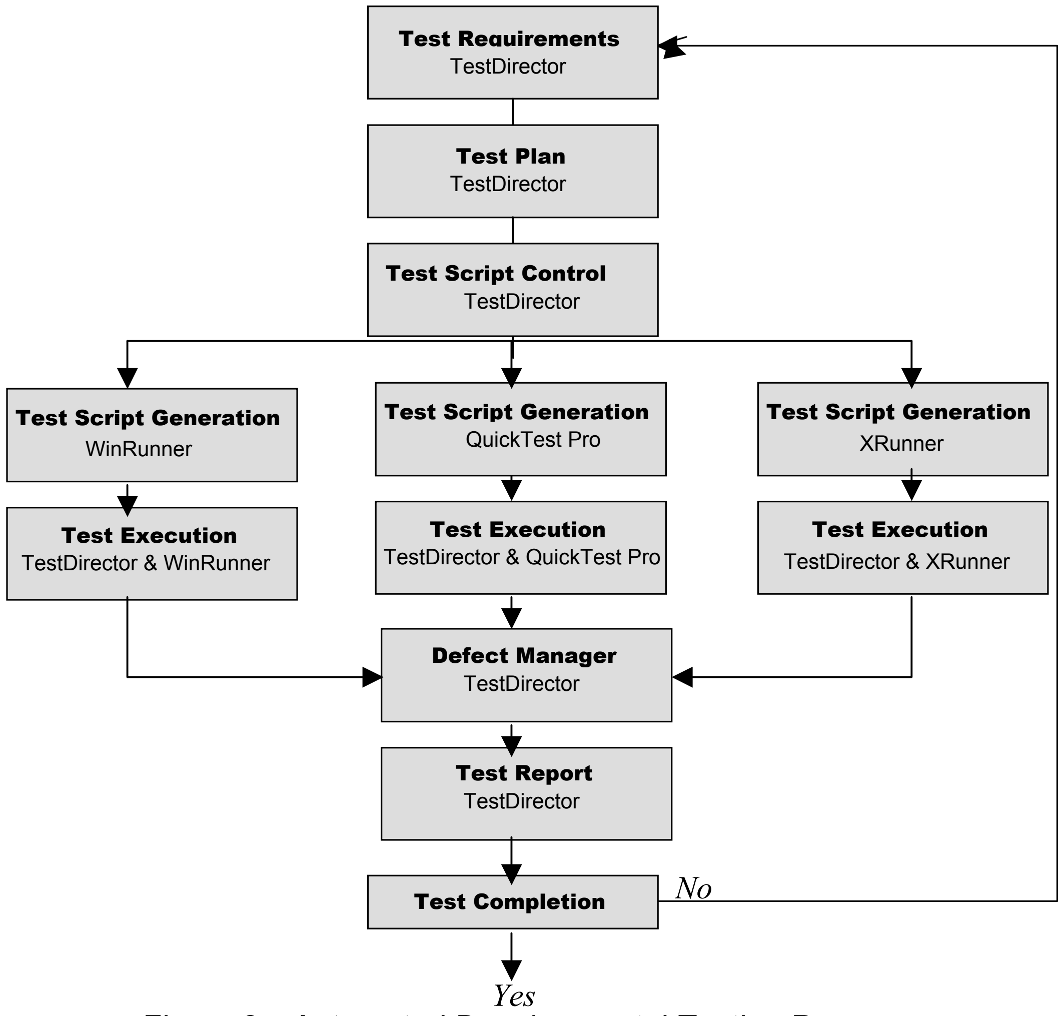


Figure 2—Automated Developmental Testing Process

Automated Operational Testing

The Automated Operational Testing follows the same process as outlined in the Automated Developmental Testing Process outlined in Figure 2. An Automated Operational Test has more complex test scripts. The additional complexity occurs as a result of programming in quality control checks. These quality control checks mimic the same quality control checks a person would perform in an Operational test, but the occur with computer speed and precision. We expect test personnel to check every screen and every database entry for correctness during an operational test. This can be very time consuming and tedious. For example the following scripts shows a three-step process to log into a C2 system and select a web link:

Automated Developmental Login	**Automated Operational Login**

# Step 1 Activate Internet Explorer 	
tl_step_once(1,0,Activate Internet Explorer);
web_browser_invoke (IE, “website URL");
wait (4);
	# Step 1 Activate Internet Explorer
	web_browser_invoke (IE, “website URL");
	If (win_exists (“Main”,4) ==E_OK)
	{
	tl_step_once(1,0, Web Activated);
	}
	else
	tl_step_once(1,1, Web Not Activated);

# Step 2 Logon 	
set_window("Main",19);
edit_set("username","tester1");
password_edit_set("password","cd0bb947e");
web_image_click("submit", 44, 8);
wait (4);
	# Step 2
	set_window("Main",19);
	edit_set("username","tester1");
	password_edit_set("password","cd0bb947e")
	;
	web_image_click("submit", 44, 8);
	
	If (win_exists (“Main1”,4) ==E_OK)
	{
	tl_step_once(2,0, Logon Successful);
	}
	else
	tl_step_once(2,2, Logon Not Successful);

# Step 3 Select SiteA From Main 1	# Step 3 Select SiteA From Main 1
tl_step_once(3,0, SiteA from Main1);	set_window("Main1",8);
set_window("Main1",8);	IF (web_link_valid("SiteA,valid)
web_link_click("SiteA");	{
wait (4);	web_link_click("SiteA");
	tl_step_once(3,0, SiteA is Available);
	}
	else
	tl_step_once(3,3, SiteA is not available);

As you can see the additional quality control checks perform the same type checks operational testers would perform; however they are performed with computer precision and accuracy. The Error statements in the AOT are more precise and are more helpful in identifying problems. Additional checks are also added to AOT scripts to replace testers analyzing databases. We use the following methodology for testing database adds, edits, and deletes that are part of an automated Command and Control System We establish the state of the database prior to running each test script. Then we determine the databases exact state once we have added records. Then we edit the records we added and determine if our edits were correct. Then we delete the same records to reestablish the exact state of the database before the each script was run. Using the powerful script development tools, in the record mode, we can either perform a Graphic Users Interface (GUI) check or a database (db) check. The GUI check looks at the page of data displayed on a user workstation and takes an image of the screen that is displayed. It compares this image with a previously saved image with accuracy to one pixel. To accomplish this we add the following lines of code to our test script where we want the GUI check to occur:

```
set_window ("main",1);
obj_check_gui ("OK", "list.ck1", gui1,1);
```

When we play back the script, the script run sand takes a checkpoint look at the screen and compares it to the saved screen in the record mode. If there is a 100% match, the script continues without displaying error information. If there is not an exact match to the pixel level and error message is generated. In this error message we receive a screen with the expected view, the view that has errors, and a view or the errors only. This helps us understand and document what went wrong.

When we record in the database method we run a SQL query of the database using Microsoft Query. The test recording software helps us log into the database itself and gives us an easy to use SQL query tool. We build our SQL query and receive our answer in a data table. The application captures the data table and compares future runs to the captured data table. The following code is added to the test script:

```
db_check ("list1.cell", "dbuf1");
```

When we play back the test script the SQL query runs in the background and compares the result data table with the data table generated in the record mode. If there are any discrepancies and error message is generated and a view of the data table with highlighted errors is displayed to help us understand what went wrong.

Automated Regression Testing Process

Traditionally about fifteen percent of all software errors are found during regression testing. It is very important not to let these errors go undetected throughout the software development process. In automated Regression Testing we use the same process we established during Automated Developmental Testing. We use the same test scripts we ran during the Automated Operation Testing. If the corrections/additions to the application require we add and or modify existing test scripts. ART can be run at anytime and requires very little human interaction. The reason this works so well is the test plan and test scripts are automated. They can be run at any time without having to bring in and train testers from all over the world to perform a controlled test. Every time a patch is put on the database or on workstations the automated regression test can be re-run and test reports can be completed in a few hours.

For example, Using ART techniques we ran a regression test for Spiral 6 TBMCS-Unit Level. The C2AT team tests scripts and test plans used during the Automated Operational Test for spiral 6 of TBMCS-Unit Level. These test scripts running on one test management workstation and two client workstations exercised each of the 224 web links. The entire test ran in about 90 minutes and the automated test report was completed and printed in about two hours.

Performance Test Process

The performance test process using automated tools is outlined in Figure 3. This process is test requirement driven. Requirements are documented in TestDirector. The number of virtual users, virtual test scripts (LoadRunner) and GUI test scripts (WinRunner, QuickTest Pro, and/or XRunner), and performance test monitoring specifics are written into the LoadRunner test plan. LoadRunner executes the test scripts. During test execution, monitoring the performance of servers, database servers, web servers, network, transactions per second, and many other key factors is accomplished. Test results are sent to the LoadRunner analyzer and then finalized in TestDirector. This capability to perform detailed testing gives us a clear indication if the software will perform within detailed performance specification. If corrections need to be made, re-running the same performance test is a routine event requiring little or no preparation time. Using these processes, developers will know how well their application performs when the maximum numbers of users fully utilize the capabilities of their workstations.

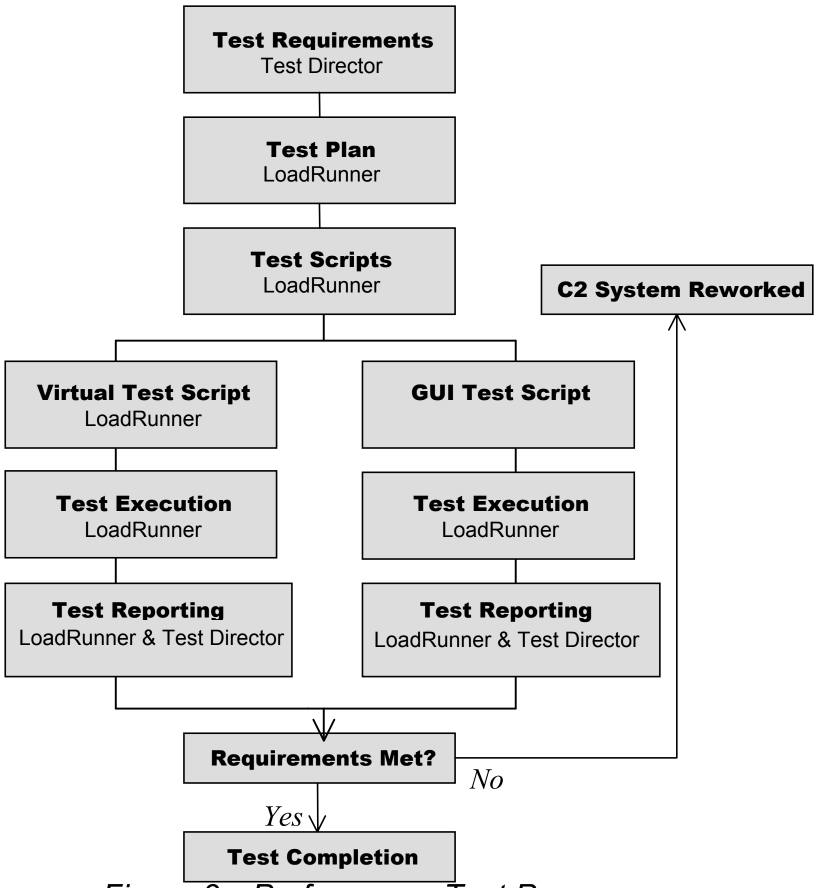


Figure 3—Performance Test Process

This process can be adapted to the specific needs of the customer. All internal computer processes can be studied to give a complete picture of system performance. In many cases, database configuration changes can be revealed that will improve system performance to meet performance specifications. If the rework necessary to meet specifications is extensive, a re-running of the automated developmental test and performance test may be required. This can be accomplished using existing automated testing.

Automated Test Management Tool

TestDirector software runs in the Microsoft Windows environment on a networked Pentium II computer with 64 MB RAM and a 1 GB Hard Drive. TestDirector has four modules inside each test plan—a Requirements module, a Test Plan module, a Test Lab (execution) module, and a Discrepancies module. TestDirector is Microsoft Word compatible and imports and generates Word documents.

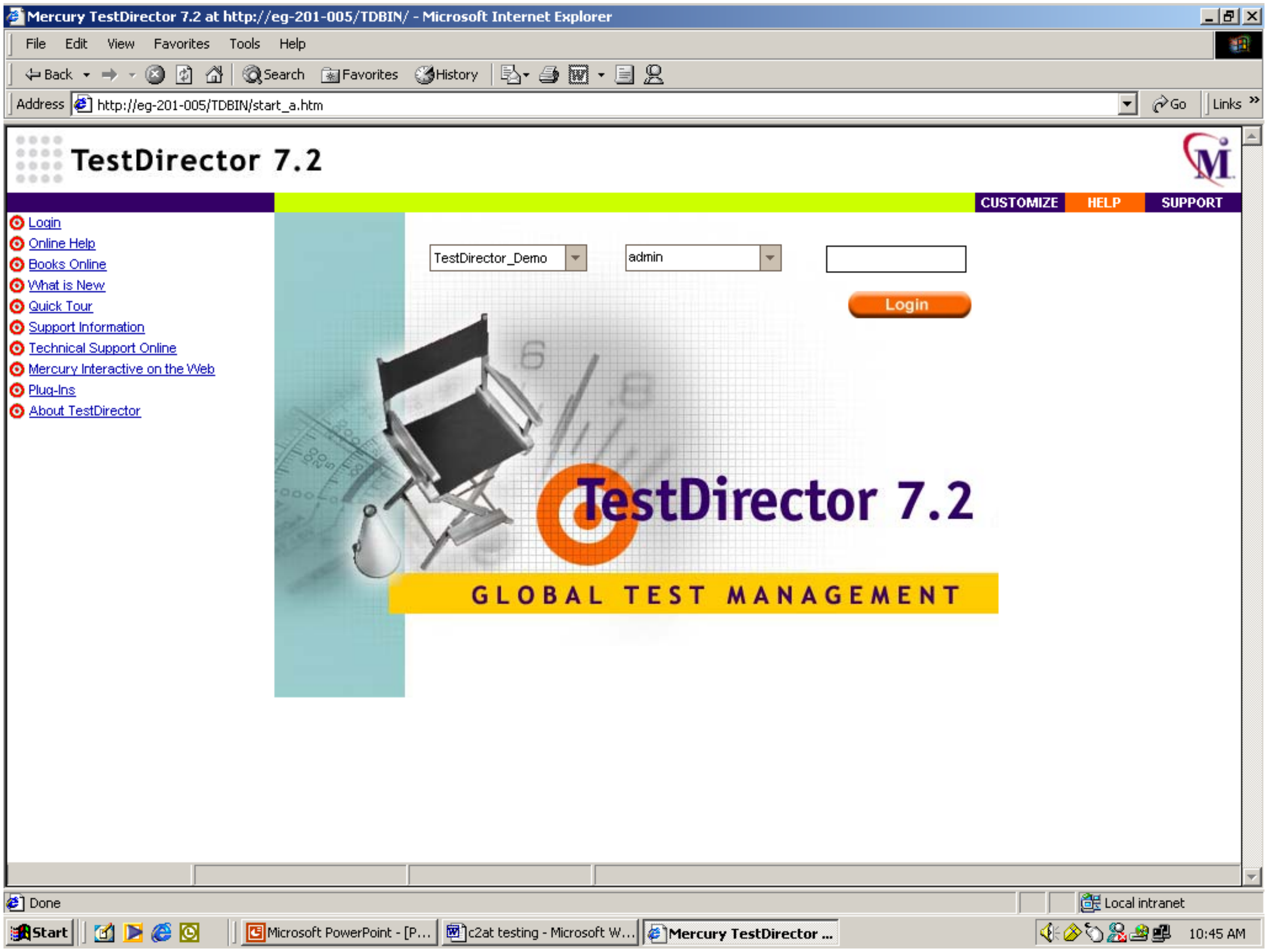
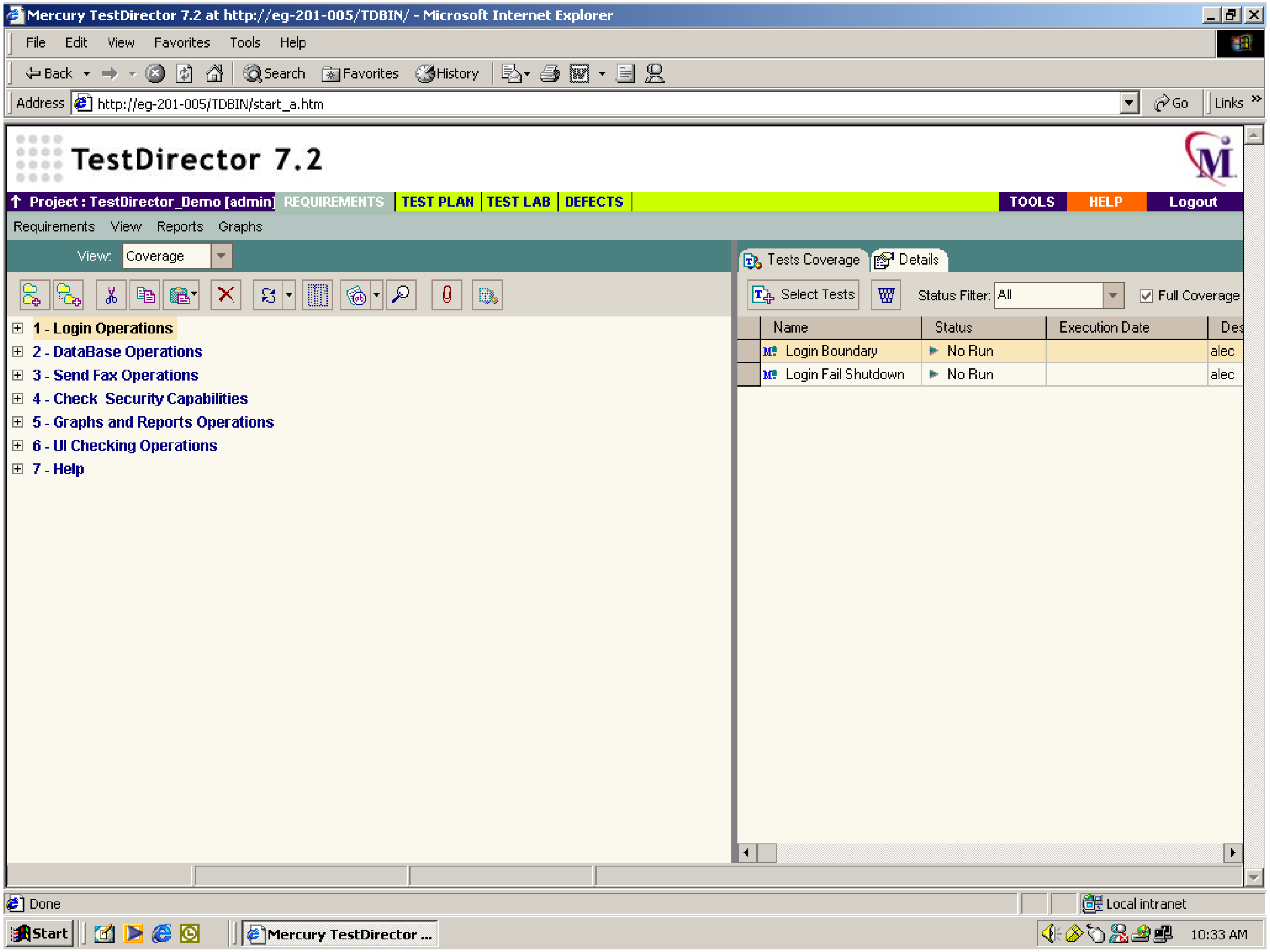


Figure 4—TestDirector

The Requirements module is a listing of all the test events that require testing. The requirements are linked to specific test scripts. The status of the current test and details about test requirements are viewed from this window. Additional descriptions of requirements can be attached to this listing via attached Microsoft Word documents.


Mercury TestDirector 7.2 at http://eg-201-005/TDBIN/ - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://eg-201-005/TDBIN/start_a.htm
Go Links
TestDirector 7.2
Project : TestDirector_Demo [admin] REQUIREMENTS TEST PLAN TEST LAB DEFECTS TOOLS HELP Logout
Requirements View Reports Graphs
Tests Coverage Details
Select Tests
Done
Local intranet
Start
Mercury TestDirector ...
10:33 AM

View: Coverage

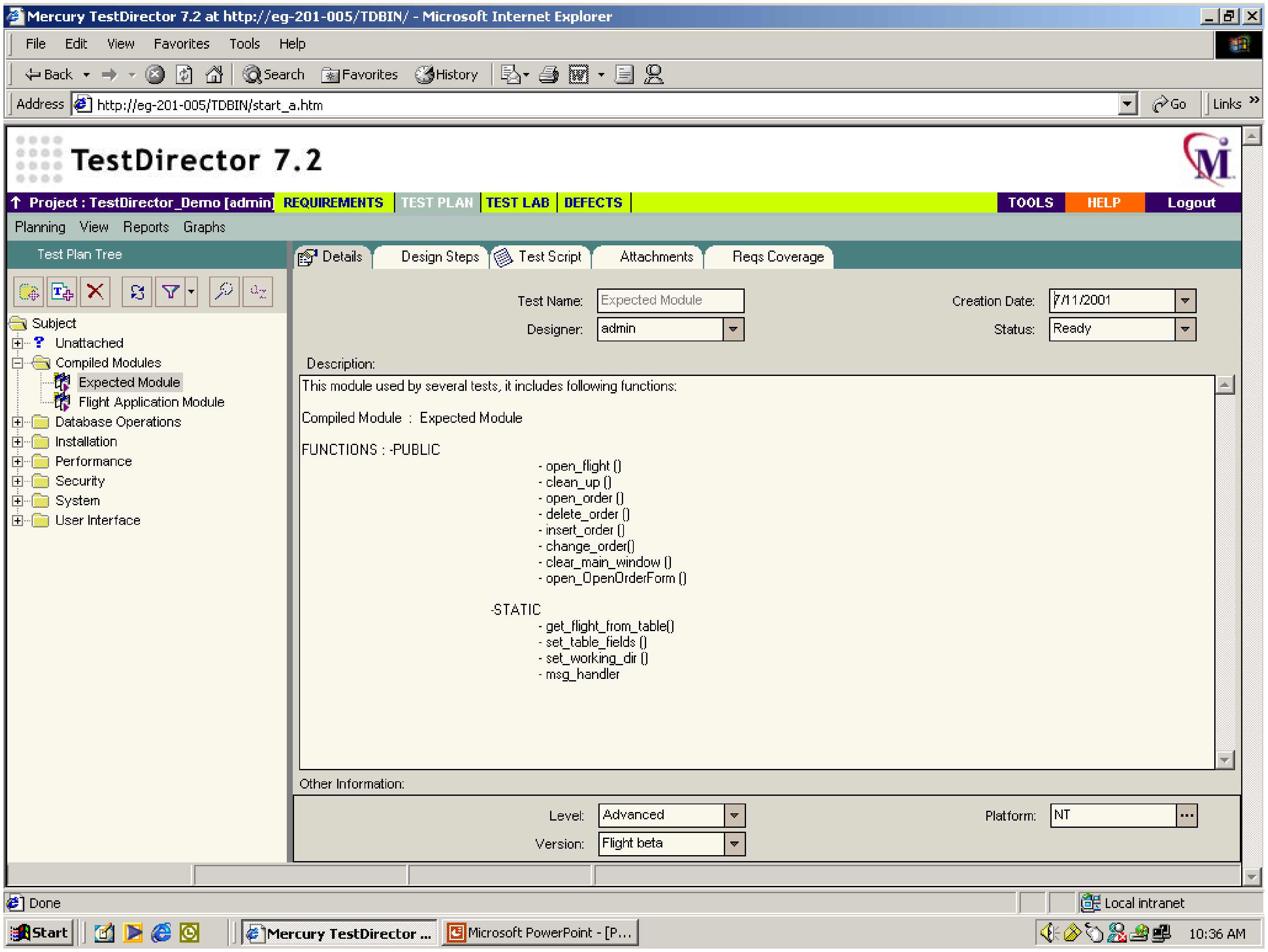
Status Filter: All ✓ Full Coverage

- 1 - Login Operations
- 2 - DataBase Operations
- 3 - Send Fax Operations
- 4 - Check Security Capabilities
- 5 - Graphs and Reports Operations
- 6 - UI Checking Operations
- 7 - Help

Name	Status	Execution Date	Des
Login Boundary	No Run		alec
Login Fail Shutdown	No Run		alec

Figure 5—TestDirector Requirements

The Test Plan module is an automated version of the standard test plans written by the 46th Test Squadron. The standard test plan can be imported/modified into the Test Plan module. Each test described in the 46th Test Squadron appendices becomes a test script. The “test card” instruction becomes steps in the TestDirector Test plan. Test Scripts following the steps in the test plan are developed using WinRunner in the Microsoft Windows environment or XRunner in the UNIX (Solaris) environment. Each test script is linked to test requirements so that testing is focused on requirements.


Mercury TestDirector 7.2 at http://eg-201-005/TDBIN/ - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://eg-201-005/TDBIN/start_a.htm
TestDirector 7.2
Project : TestDirector_Demo [admin]
REQUIREMENTS
TEST PLAN
TEST LAB
DEFECTS
TOOLS
HELP
Logout
Planning View Reports Graphs
Test Plan Tree
Details
Design Steps
Test Script
Attachments
Reqs Coverage
Subject
Unattached
Compiled Modules
Expected Module
Flight Application Module
Database Operations
Installation
Performance
Security
System
User Interface

Test Name: Expected Module
Designer: admin
Creation Date: 7/11/2001
Status: Ready

Description:

This module used by several tests, it includes following functions:

Compiled Module : Expected Module

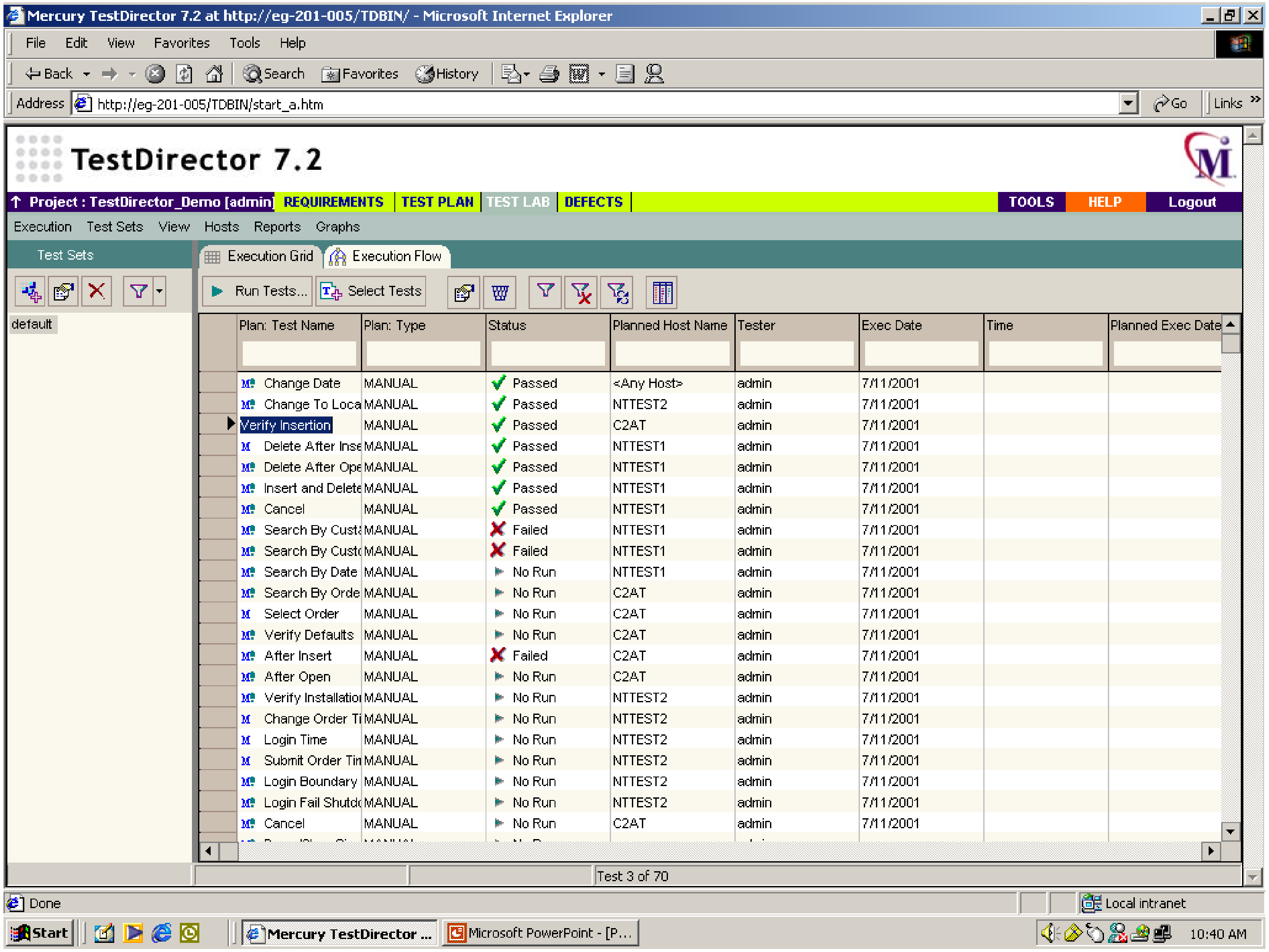
```
FUNCTIONS : -PUBLIC
                    - open_flight ()
                    - clean_up ()
                    - open_order ()
                    - delete_order ()
                    - insert_order ()
                    - change_order()
                    - clear_main_window ()
                    - open_OpenOrderForm ()

          -STATIC
                    - get_flight_from_table()
                    - set_table_fields ()
                    - set_working_dir ()
                    - msg_handler
```

Other Information:

Level: Advanced
Version: Flight beta
Platform: NT

Figure 6—TestDirector Test Plan

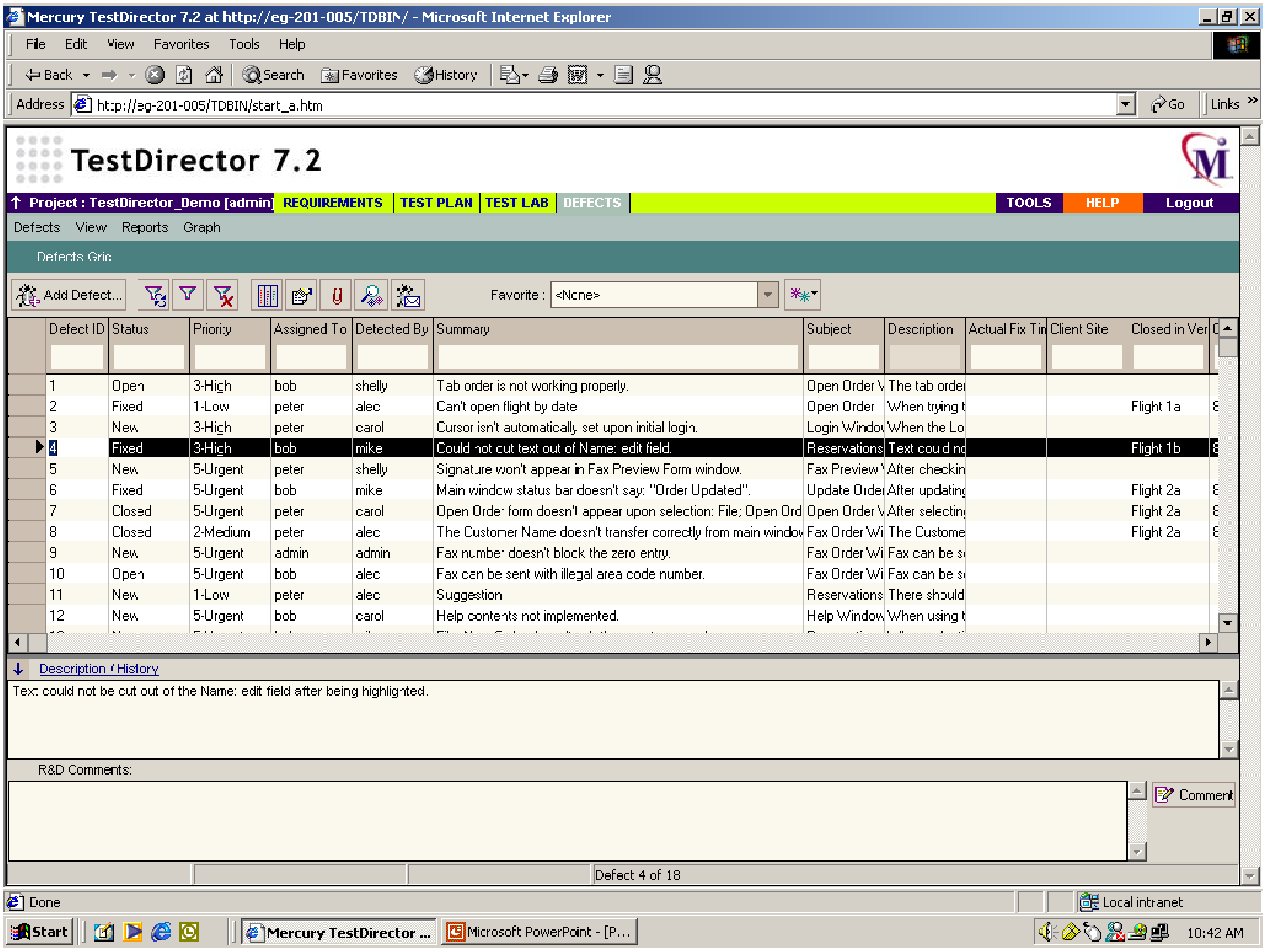
The Test Lab executes the test in a specified manner and links test script pass/fail to each specified requirement. Test scripts can be executed at a certain time, in a prescribed order, randomly, and on specific workstations or randomly selected workstations. Test execution can be designed in the Test Lab to meet the needs of the test customer.


Mercury TestDirector 7.2 at http://eg-201-005/TDBIN/ - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://eg-201-005/TDBIN/start_a.htm
Go Links
TestDirector 7.2
Project : TestDirector_Demo [admin]
REQUIREMENTS
TEST PLAN
TEST LAB
DEFECTS
TOOLS
HELP
Logout
Execution Test Sets View Hosts Reports Graphs
Test Sets
Execution Grid
Execution Flow
Run Tests...
Select Tests
default
Test 3 of 70
Done
Local intranet
Start
Mercury TestDirector ...
Microsoft PowerPoint - [P...
10:40 AM

Plan: Test Name	Plan: Type	Status	Planned Host Name	Tester	Exec Date	Time	Planned Exec Date
Change Date	MANUAL	✓ Passed	<Any Host>	admin	7/11/2001		
Change To Loca	MANUAL	✓ Passed	NTTEST2	admin	7/11/2001		
Verify Insertion	MANUAL	✓ Passed	C2AT	admin	7/11/2001		
Delete After Inse	MANUAL	✓ Passed	NTTEST1	admin	7/11/2001		
Delete After Ope	MANUAL	✓ Passed	NTTEST1	admin	7/11/2001		
Insert and Delete	MANUAL	✓ Passed	NTTEST1	admin	7/11/2001		
Cancel	MANUAL	✓ Passed	NTTEST1	admin	7/11/2001		
Search By Custo	MANUAL	✗ Failed	NTTEST1	admin	7/11/2001		
Search By Custo	MANUAL	✗ Failed	NTTEST1	admin	7/11/2001		
Search By Date	MANUAL	▸ No Run	NTTEST1	admin	7/11/2001		
Search By Orde	MANUAL	▸ No Run	C2AT	admin	7/11/2001		
Select Order	MANUAL	▸ No Run	C2AT	admin	7/11/2001		
Verify Defaults	MANUAL	▸ No Run	C2AT	admin	7/11/2001		
After Insert	MANUAL	✗ Failed	C2AT	admin	7/11/2001		
After Open	MANUAL	▸ No Run	C2AT	admin	7/11/2001		
Verify Installatio	MANUAL	▸ No Run	NTTEST2	admin	7/11/2001		
Change Order Ti	MANUAL	▸ No Run	NTTEST2	admin	7/11/2001		
Login Time	MANUAL	▸ No Run	NTTEST2	admin	7/11/2001		
Submit Order Tin	MANUAL	▸ No Run	NTTEST2	admin	7/11/2001		
Login Boundary	MANUAL	▸ No Run	NTTEST2	admin	7/11/2001		
Login Fail Shutdo	MANUAL	▸ No Run	NTTEST2	admin	7/11/2001		
Cancel	MANUAL	▸ No Run	C2AT	admin	7/11/2001		

Figure 7—TestDirector Test Lab

Test Script failures are identified and placed in the Discrepancy module. The Discrepancy module helps identify, prioritize, and assign responsibility to each error that occurs during testing. A complete test report is generated as a Microsoft Word document, which can be edited if necessary. Recently we ran a 224 script developmental test and the test report completed in less than four hours.


Mercury TestDirector 7.2 at http://eg-201-005/TDBIN/ - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://eg-201-005/TDBIN/start_a.htm Go Links
TestDirector 7.2
Project : TestDirector_Demo [admin] REQUIREMENTS TEST PLAN TEST LAB DEFECTS TOOLS HELP Logout
Defects View Reports Graph
Defects Grid
Add Defect...
Favorite : <None>
Description / History
R&D Comments:
Comment
Defect 4 of 18
Done
Local intranet
Start
Mercury TestDirector ...
Microsoft PowerPoint - [P...
10:42 AM

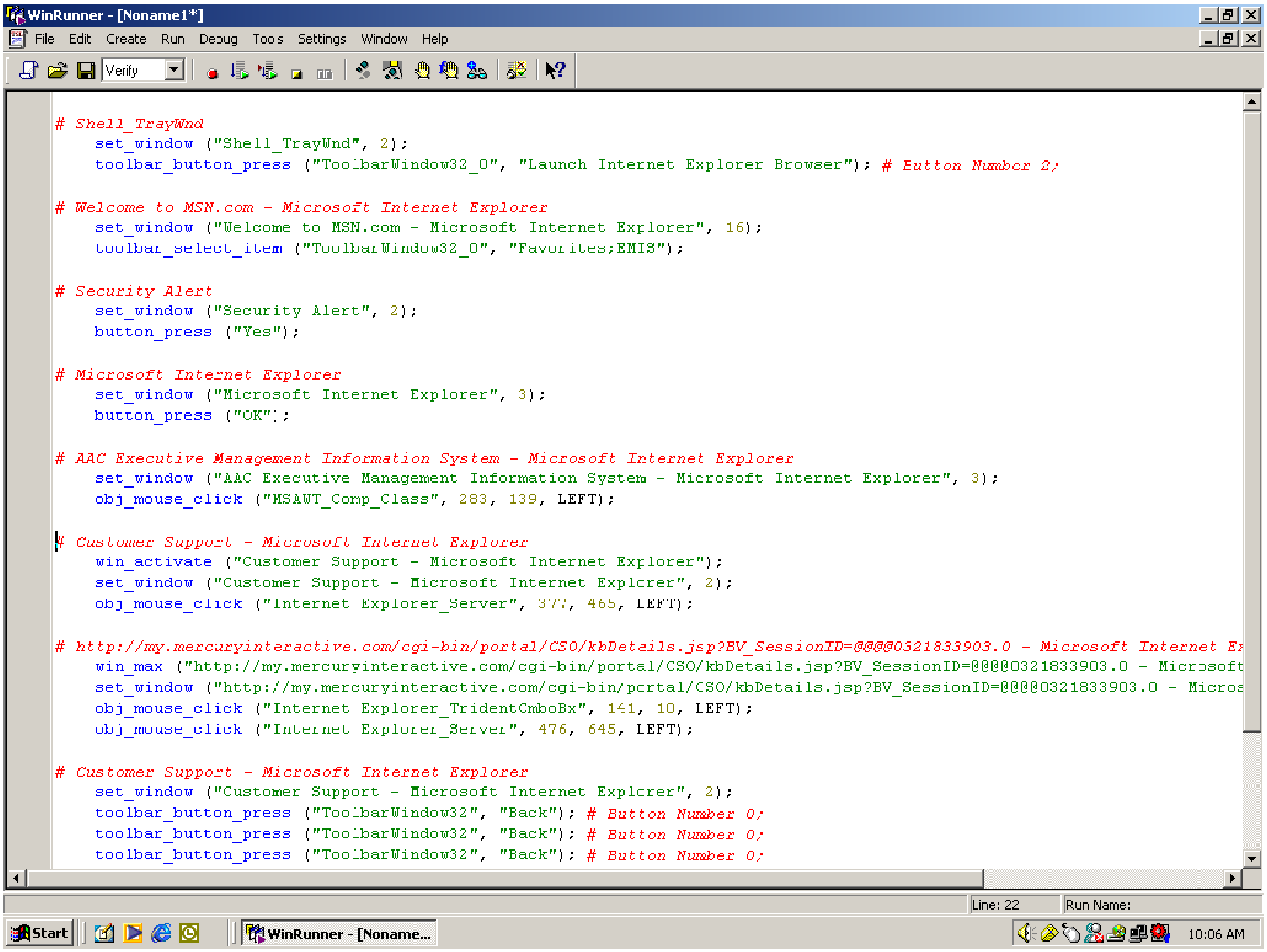
Defect ID	Status	Priority	Assigned To	Detected By	Summary	Subject	Description	Actual Fix Tin	Client Site	Closed in Ver
1	Open	3-High	bob	shelly	Tab order is not working properly.	Open Order V	The tab order			
2	Fixed	1-Low	peter	alec	Can't open flight by date	Open Order	When trying t			Flight 1a
3	New	3-High	peter	carol	Cursor isn't automatically set upon initial login.	Login Windov	When the Lo			
4	Fixed	3-High	bob	mike	Could not cut text out of Name: edit field.	Reservations	Text could no			Flight 1b
5	New	5-Urgent	peter	shelly	Signature won't appear in Fax Preview Form window.	Fax Preview V	After checkin			
6	Fixed	5-Urgent	bob	mike	Main window status bar doesn't say: "Order Updated".	Update Order	After updating			Flight 2a
7	Closed	5-Urgent	peter	carol	Open Order form doesn't appear upon selection: File; Open Ord	Open Order V	After selecting			Flight 2a
8	Closed	2-Medium	peter	alec	The Customer Name doesn't transfer correctly from main window	Fax Order Wi	The Custome			Flight 2a
9	New	5-Urgent	admin	admin	Fax number doesn't block the zero entry.	Fax Order Wi	Fax can be s			
10	Open	5-Urgent	bob	alec	Fax can be sent with illegal area code number.	Fax Order Wi	Fax can be s			
11	New	1-Low	peter	alec	Suggestion	Reservations	There should			
12	New	5-Urgent	bob	carol	Help contents not implemented.	Help Window	When using t			

Text could not be cut out of the Name: edit field after being highlighted.

Figure 8—TestDirector Defects Manager

ADT/AOT/ART Tools

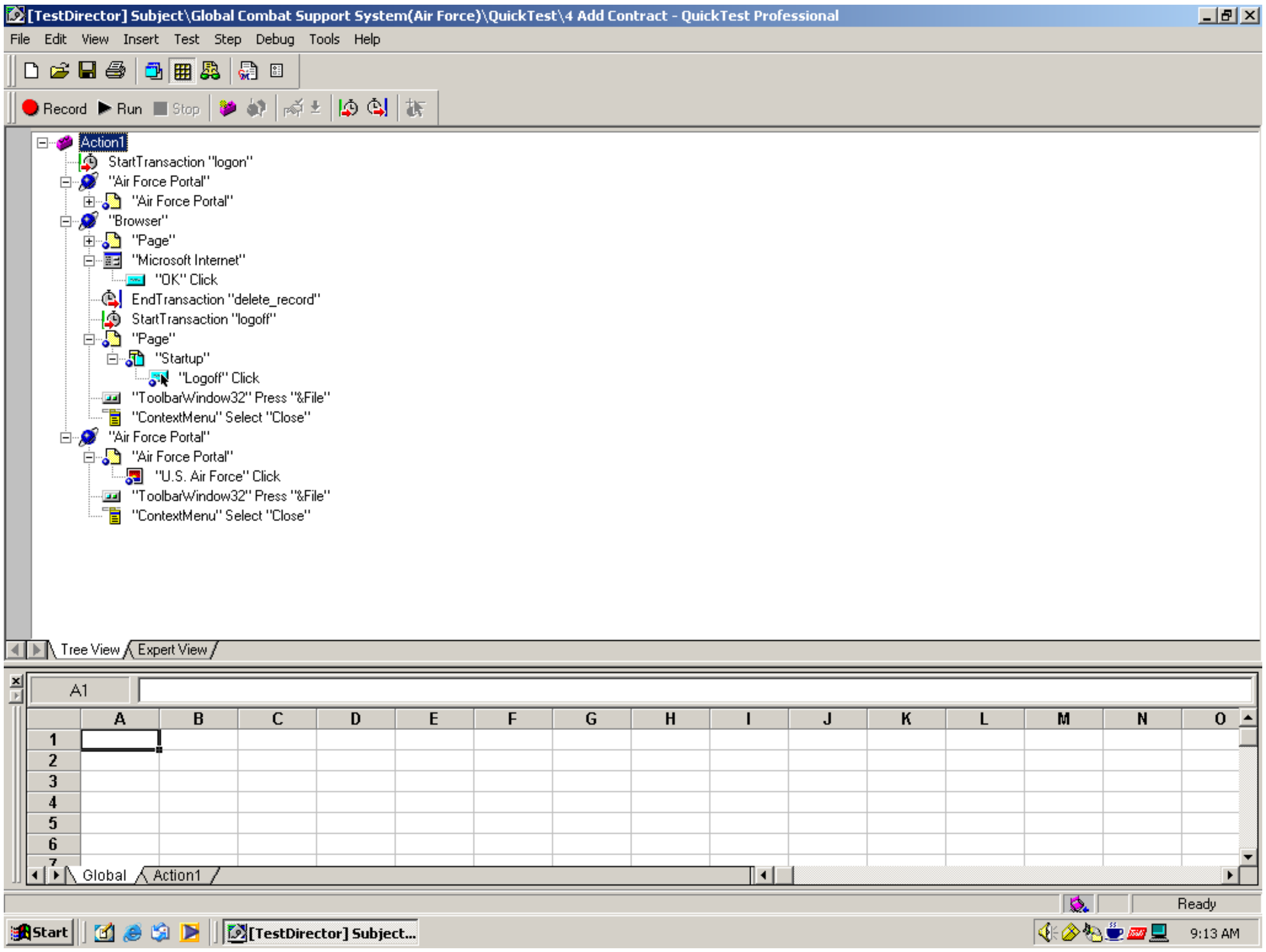
WinRunner software runs in the Microsoft Windows environment on a networked Pentium III computer with 128 MB RAM and a 10 GB Hard Drive. WinRunner is installed on the same workstation as your application. Each "test card" is executed on a user workstation and captured by Win Runner's record capability. Recorded inputs are saved as test scripts. Database checks and user screen integrity checks can be added to recorded scripts to improve their value. The test scripts are saved in TestDirector and played back as part of an automated test. WinRunner uses Test Script Language, which is similar to C++ for recording scripts and playing back scripts. Script can be saved in TestDirector and played back using the test lab execution feature in TestDirector.



```
# Shell_TrayWnd
    set_window ("Shell_TrayWnd", 2);
    toolbar_button_press ("ToolbarWindow32_0", "Launch Internet Explorer Browser"); # Button Number 2;

# Welcome to MSN.com - Microsoft Internet Explorer
    set_window ("Welcome to MSN.com - Microsoft Internet Explorer", 16);
    toolbar_select_item ("ToolbarWindow32_0", "Favorites;EMIS");

# Security Alert
    set_window ("Security Alert", 2);
    button_press ("Yes");

# Microsoft Internet Explorer
    set_window ("Microsoft Internet Explorer", 3);
    button_press ("OK");

# AAC Executive Management Information System - Microsoft Internet Explorer
    set_window ("AAC Executive Management Information System - Microsoft Internet Explorer", 3);
    obj_mouse_click ("MSAWT_Comp_Class", 283, 139, LEFT);

# Customer Support - Microsoft Internet Explorer
    win_activate ("Customer Support - Microsoft Internet Explorer");
    set_window ("Customer Support - Microsoft Internet Explorer", 2);
    obj_mouse_click ("Internet Explorer_Server", 377, 465, LEFT);

# http://my.mercuryinteractive.com/cgi-bin/portal/CSO/kbDetails.jsp?BV_SessionID=@@@@0321833903.0 - Microsoft Internet E
    win_max ("http://my.mercuryinteractive.com/cgi-bin/portal/CSO/kbDetails.jsp?BV_SessionID=@@@@0321833903.0 - Microsoft
    set_window ("http://my.mercuryinteractive.com/cgi-bin/portal/CSO/kbDetails.jsp?BV_SessionID=@@@@0321833903.0 - Micros
    obj_mouse_click ("Internet Explorer_TridentCmboBx", 141, 10, LEFT);
    obj_mouse_click ("Internet Explorer_Server", 476, 645, LEFT);

# Customer Support - Microsoft Internet Explorer
    set_window ("Customer Support - Microsoft Internet Explorer", 2);
    toolbar_button_press ("ToolbarWindow32", "Back"); # Button Number 0;
    toolbar_button_press ("ToolbarWindow32", "Back"); # Button Number 0;
    toolbar_button_press ("ToolbarWindow32", "Back"); # Button Number 0;
```

Figure 9—WinRunner

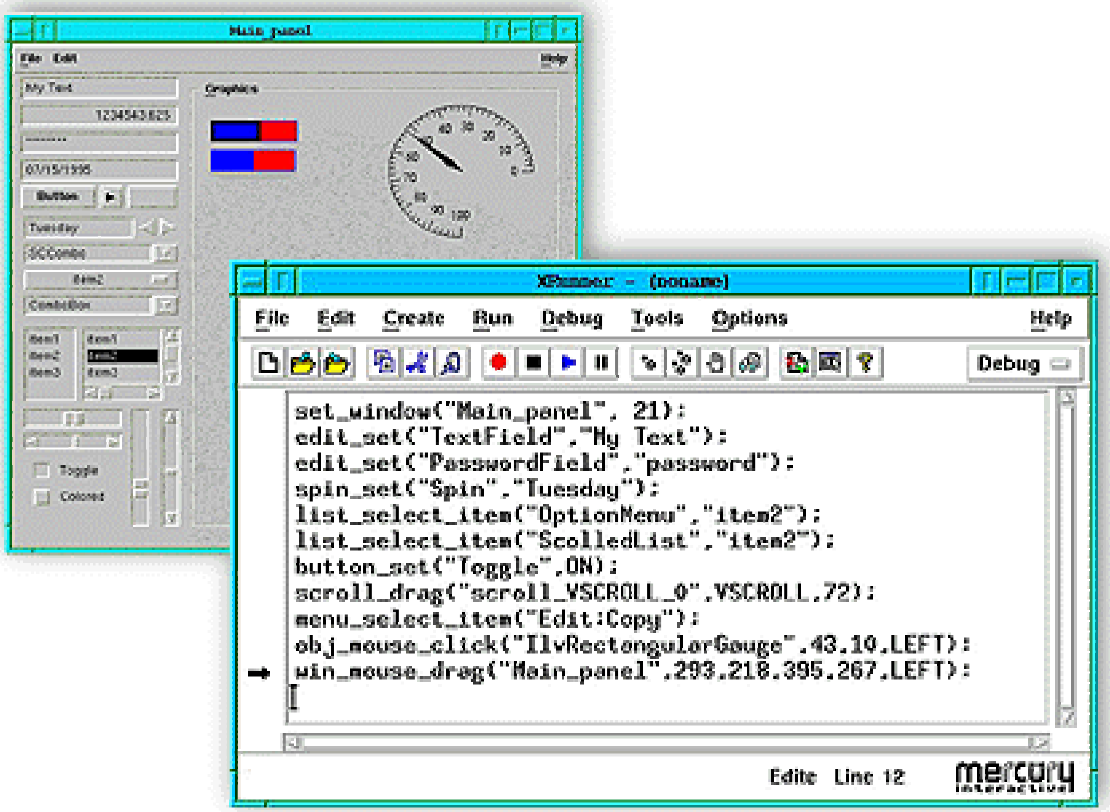
QuickTest Pro software runs in the web development environment on a networked Pentium III computer with 128 MB RAM and a 10 GB Hard Drive. QuickTest Pro is installed on the same workstation as your application. Each “test card” is executed on a user workstation and captured by QuickTest Pro’s record capability. Recorded inputs are saved as test scripts. Database checks and user screen integrity checks can be added to recorded scripts to improve their value. The test scripts are saved in TestDirector and played back as part of an automated test. QuickTest Pro uses Microsoft Visual Basic as its test script language for recording scripts and playing back scripts. Script can be saved in TestDirector and played back using the test lab execution feature in TestDirector.


[TestDirector] Subject\Global Combat Support System(Air Force)\QuickTest\4 Add Contract - QuickTest Professional
File Edit View Insert Test Step Debug Tools Help
Record Run Stop
Tree View Expert View
A1
A B C D E F G H I J K L M N O
1 2 3 4 5 6 7
Global Action1
Ready
Start
[TestDirector] Subject...
9:13 AM

```
Action1
    StartTransaction "logon"
    "Air Force Portal"
        "Air Force Portal"
    "Browser"
        "Page"
        "Microsoft Internet"
            "OK" Click
        EndTransaction "delete_record"
        StartTransaction "logoff"
        "Page"
            "Startup"
                "Logoff" Click
        "ToolbarWindow32" Press "&File"
        "ContextMenu" Select "Close"
    "Air Force Portal"
        "Air Force Portal"
            "U.S. Air Force" Click
        "ToolbarWindow32" Press "&File"
        "ContextMenu" Select "Close"
```

Figure 10—QuickTest Pro

XRunner is the "x-windows" UNIX equivalent of WinRunner. XRunner runs on HP-UX 11, generic UNIX, and Solaris 2.51 or higher. A minimum of 500 MB of storage plus 5 MB per test script is required. 64 MB RAM is the minimum memory requirement. Each "test card" when executed by an XRunner user, is recorded and saved for playback as part of an automated test. Database checks and user screen integrity checks can be added to recorded scripts to improve their value. These scripts are saved in TestDirector and can be played back by TestDirector as part of automated testing.

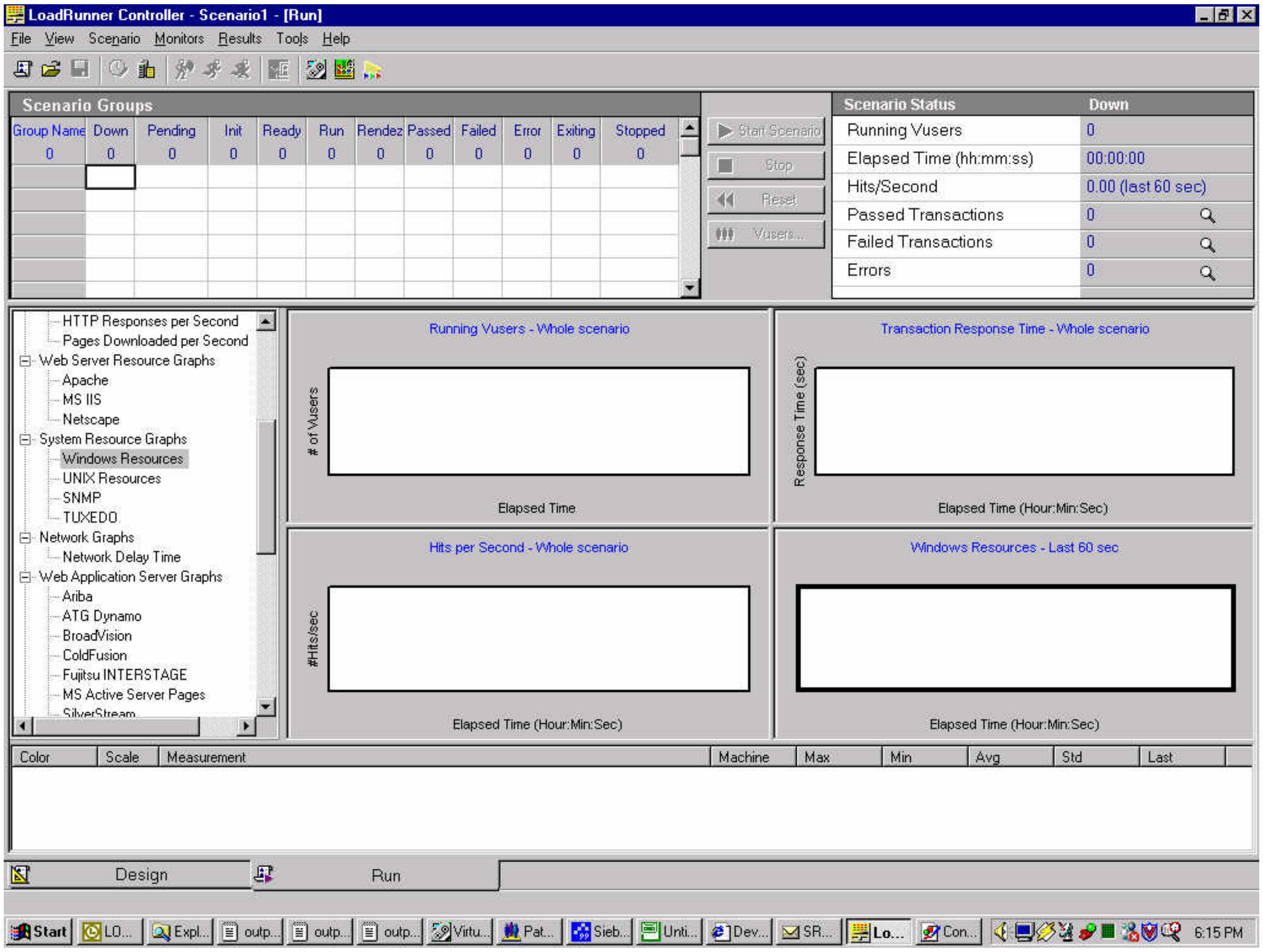


```
set_window("Main_panel", 21);
edit_set("TextField","My Text");
edit_set("PasswordField","password");
spin_set("Spin","Tuesday");
list_select_item("OptionMenu","item2");
list_select_item("ScolledList","item2");
button_set("Toggle",ON);
scroll_drag("scroll_VSCROLL_0",VSCROLL,72);
menu_select_item("Edit:Copy");
obj_mouse_click("IlvRectangularGauge",43,10,LEFT);
win_mouse_drag("Main_panel",293,218,395,267,LEFT);
```

Figure 11—XRunner

Automated Performance Testing Tool

LoadRunner runs in the Microsoft Windows, UNIX, HP-UX, Solaris, and LINUX environments. Current user licenses allow 250 virtual users, 250 web users, 5 GUI users (WinRunner and XRunner), Server Monitor (Microsoft Windows, UNIX, Solaris, HP-UX, and LINUX), Network Delay Monitor, Oracle Monitor, SQL Server Monitor, and web server monitors. With LoadRunner we can simulate heavy user loads to monitor server performance, network performance, and system performance. This tool provides a complete picture of the entire system under stress with minimal preparation for testing time required. Complete test script work up can usually be accomplished in less than one day. Detailed database server monitoring determines how well the relational database management system performs under the full load that the system is designed to accommodate. We recently ran a performance test that included 130,000 transactions in a two-hour period performed by 100 virtual users. With the enhanced monitoring capabilities we have we were able to determine whom well the entire system performed under the prescribed load.


LoadRunner Controller - Scenario1 - [Run]
File View Scenario Monitors Results Tools Help
Scenario Groups
Start Scenario
Stop
Reset
Vusers...
HTTP Responses per Second
Pages Downloaded per Second
Web Server Resource Graphs
Apache
MS IIS
Netscape
System Resource Graphs
Windows Resources
UNIX Resources
SNMP
TUXEDO
Network Graphs
Network Delay Time
Web Application Server Graphs
Ariba
ATG Dynamo
BroadVision
ColdFusion
Fujitsu INTERSTAGE
MS Active Server Pages
SilverStream
Running Vusers - Whole scenario
# of Vusers
Elapsed Time
Transaction Response Time - Whole scenario
Response Time (sec)
Elapsed Time (Hour:Min:Sec)
Hits per Second - Whole scenario
#Hits/sec
Elapsed Time (Hour:Min:Sec)
Windows Resources - Last 60 sec
Elapsed Time (Hour:Min:Sec)
Color Scale Measurement Machine Max Min Avg Std Last
Design
Run
Start LO... Expl... outp... outp... outp... Virtu... Pat... Sieb... Unti... Dev... SR... Lo... Con... 6:15 PM

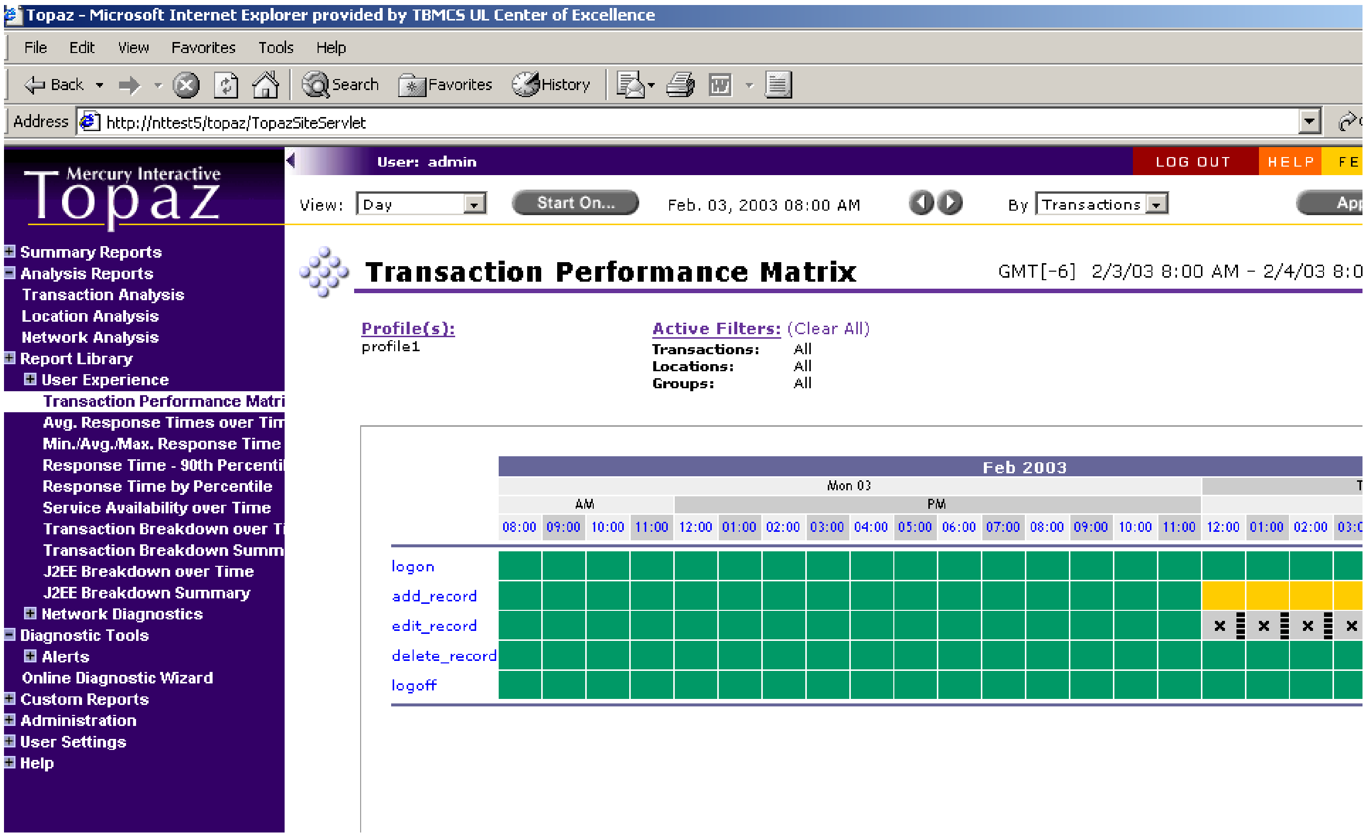
Group Name	Down	Pending	Init	Ready	Run	Rendez	Passed	Failed	Error	Exiting	Stopped
0	0	0	0	0	0	0	0	0	0	0	0

Scenario Status	Down
Running Vusers	0
Elapsed Time (hh:mm:ss)	00:00:00
Hits/Second	0.00 (last 60 sec)
Passed Transactions	0
Failed Transactions	0
Errors	0

Figure 12—LoadRunner Controller

Automated 24/7 Monitoring Tool

Topaz runs in the Microsoft Windows, UNIX, HP-UX, Solaris, and LINUX environments. The Topaz website workstation run on Pentium III PCs with 256 MB Ram Current and 10 GB HD under MS Windows 2000 Server with Microsoft SQL Server 2000 database. Our current user licenses allow 25 user defined business transactions. Topaz user workstations (Microsoft Windows, UNIX, HP-UX, Solaris, and LINUX) generate test scripts using Astra QuickTest (similar to QuickTest Pro). Monitoring conditions, early warning conditions, and notification procedures are setup in the Topaz Controller (Pentium III PC w/ 128 MB RAM, 10GB HD, running Windows 2000). Topaz can passively monitor the entire Command and Control Information and provide early warning of problems with either e-mail or pager alerts without running a process on any server in the network. The Topaz web site keeps a repository of performance data in a Microsoft SQL Server database. Using the power of the Topaz interface we can drill down using Topaz's easy to use website and determine the cause of problems if they occur. We are able to specifically monitor every major database (DB2, Sybase, Oracle, Informix, and SQL Server), server (Windows, Unix, and Linux), web-server, the local area network, beyond the firewall, and link to other Open Test Architecture (OTA) compliant monitors such as BMC patrol and Tivoli.


Topaz - Microsoft Internet Explorer provided by TBMCS UL Center of Excellence
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://nttest5/topaz/TopazSiteServlet
Mercury Interactive Topaz
User: admin
LOG OUT HELP FE
View: Day
Start On...
Feb. 03, 2003 08:00 AM
By Transactions
App
Summary Reports
Analysis Reports
Transaction Analysis
Location Analysis
Network Analysis
Report Library
User Experience
Transaction Performance Matri
Avg. Response Times over Tim
Min./Avg./Max. Response Time
Response Time - 90th Percenti
Response Time by Percentile
Service Availability over Time
Transaction Breakdown over Ti
Transaction Breakdown Summ
J2EE Breakdown over Time
J2EE Breakdown Summary
Network Diagnostics
Diagnostic Tools
Alerts
Online Diagnostic Wizard
Custom Reports
Administration
User Settings
Help

Transaction Performance Matrix GMT[-6] 2/3/03 8:00 AM - 2/4/03 8:0

Profile(s):
profile1

Active Filters: (Clear All)

Transactions:	All
Locations:	All
Groups:	All

	Feb 2003																			
	Mon 03																T			
	AM				PM															
	08:00	09:00	10:00	11:00	12:00	01:00	02:00	03:00	04:00	05:00	06:00	07:00	08:00	09:00	10:00	11:00	12:00	01:00	02:00	03:0
logon	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
add_record	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟨	🟨	🟨	🟨
edit_record	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	×	×	×	×
delete_record	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
logoff	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

Figure 13—Topaz Web Site

Future Development

The Freeman Computer Sciences Center is committed to providing our customers the best automated developmental testing of computerized Command and Control systems. Once developmental testing is complete, we can load test the application to determine how well it performs with the maximum number of users it is designed to service. All future automated command and control systems can be tested in this manner. Better and more complete testing will lead to faster and more effective system development.

Conclusion

C2AT is the most effective way to perform automated developmental testing, automated operational testing, automated regression testing, automated performance testing and automated 24/7 monitoring of automated command and control systems. Currently 75% of the Fortune 500 companies in the United States use these same tools to test their automated information systems. Using the power of automated tools to perform ADT, AOT, ART, APT, and A24/7SM gives a complete testing framework to a Computerized Command and Control system. Automated tools cannot answer the questions plan: Does the user like it? Does the application meet the user’s needs? Other than these questions automated testing is more accurate and complete than human testing of C2 systems. It frees up people to better answer the two important questions mentioned above. Increased testing effectiveness helps speed up the deployment time and reduces risk associated with deploying systems that have not been properly tested. This is especially true for applications that are developed using the spiral development process model.

To request current information on our C2AT capabilities; please see the Contact List at the front of this document.

Command and Control Automated Testing Capabilities (C2AT)

Information Request

QUESTIONS:

COMMENTS/SUGGESTIONS:

NAME:	**MAILING ADDRESS:**
PHONE:	
FAX:	
INTERNET:	
DATE SUBMITTED:	

PLEASE ROUTE TO:

Freeman Computer Sciences Center

Attn: Test and Analysis Division Customer Service

201 W. Eglin Blvd. Suite 228

Eglin AFB FL 32542–6829

FAX: (850) 882–5924

2653.1