Performance Validation and Auto-method Test Management

Elena de Castro Díaz-Plaza

Master of Science Thesis
Stockholm, Sweden 2006

ICT/ECS-2006-44

Performance Validation and Auto-method Test Management

Elena de Castro Díaz-Plaza

Examiner
Dr. Christian Schulte

Master of Science Thesis
Stockholm, Sweden 2006

ICT/ECS-2006-44

Background

Automated tests methodologies applied to test applications are based on scripts that emulate one user activity or multiple at the same time depending on whether the application accepts multiple users or not.

There are different types of tools to help with the script creation depending on the type of application tested and the goal pursued. For example, there are different tools to test an interface or to generate a huge load of activity for an application.

Besides, due to the increasing complexity of today's applications and the competitive pressures and costs of application failure, the demand for powerful test management is even greater. Since one has to handle with many projects, the best approach is adopting a good test management methodology that helps to centralize, organize, prioritize and document the tests to ensure coverage of the requirements, consistency and reusability of the tests campaigns

Abstract

The work described in this thesis was carried out at the Amadeus Development Site, Nice, France. The work encompasses the completion of two separate projects.

SPRING project

SPRING project studies the possibility of using Microsoft Project Server 2003 (MPS 2003) at Amadeus.
Prior to the completion of the SPRING project, Amadeus used a set of logically separate applications to perform project management duties and manage available resources.

The Microsoft Project Server 2003 application provides planning and resource management under a collaboration model. An evaluation has shown that the tool reaches an acceptable level of functionality and capacity, to cope with the requirements of Amadeus.

The SPRING project will investigate the performance, reliability and scalability of the MPS 2003 in the following context:

- Scenarios exercising “time reporting” and “project plan management” activities
- Production infrastructure (application and database servers) hosted in Nice
- Two different client access types: from LAN workstations and from a Terminal server machine

The analysis of the test results provide evidence that there is no risk in terms of performance and capacity by deploying the SPRING project. However, a database management problem was detected due to the higher load, resulting in a greatly increased response time.

TestDirector migration, feasibility study and customisation

The Internet and Front Office Quality team (IFQ team) in Amadeus needed to update the platform used for testing management.

The project performed a feasibility study of the migration to a new test management platform. The study took into account the old platform features and the team requirements which permitted customisation to the specific needs of the team.

Upon completion of the feasibility study, the test management tool selected was TestDirector. The latest version includes many new customisation possibilities which have made many companies choose this product. The project explored all of the customisation possibilities and matched against the requirements of the IFQ team to decide if migration was possible.

The project concludes that the migration is possible and all the features have been implemented in the test management platform. TestDirector will be used by the IFQ team and its use is likely to be extended to many other teams within Amadeus.

Table of contents

1) Introduction

This thesis is about auto test methods and new test management solutions. To start, it gives an overview of the technologies developed by the most important companies in the testing market and an introduction of the selected tools to test interfaces, server responsiveness, generation of virtual users and test management.

This thesis is performed in Amadeus Global Distributed System Corporation and it is focused in solving two specific problems but both within the testing field. That is why first, it tries to get the reader familiar with the environment and afterwards, it refers to the two projects: "SPRING project" and "TestDirector migration, feasibility study and customisation". It ends with some general conclusions, acknowledgments, bibliography and an appendix for further information in technical details.

2) Amadeus

Amadeus is a global distribution system (GDS) and technology provider focused on the marketing, sales and distribution needs of the world's travel and tourism industries. Amadeus technology is used by airlines not only for distribution, but also for bookings at the airline's airport ticket offices (ATOs) and city ticket offices (CTOs). In addition, Amadeus provides essential sales tools for travel agents and also allows business travelers to book arrangements from their computer.

Through the Amadeus GDS, travel agencies and airline offices can book the 95 per cent of the world's scheduled airline seats. The rapid expansion of this business has made Amadeus becoming the fastest growing and most widely used global distribution system.

.

3) Related work

Depending on the author one can find many kinds of testing. Here an introduction on the basics of testing will be explained so that it helps to the comprehension of the rest of the dissertation.

Automated tests methodologies applied to test applications are based on scripts that emulate one user activity or multiple at the same time depending on whether the application accepts multiple users or not.

There are different types of tools to help with the script creation depending on the type of application tested and the goal pursued. For example, there are different tools to test an interface or to generate a huge load of activity for an application

Steps for automated testing:

- Define the testing goals
- Define the business process
- Define what kind of testing to apply :

    - Functional Tests validate the system's logical and business functions. It applies inputs to a program that matches its specifications and studies the corresponding outputs independently of the internal logic of the program.

o Performance Tests evaluate if a system or software match some specific performance requirements, such as response time, resource utilization and transaction rates.

o Load Tests evaluate systems load capacity under various operating conditions. It checks the system limits in terms of number of users.

- ♦ Define the Test Platform.
- ♦ Choose the right tool to generate the scripts
- ♦ Generate the scripts.
- ♦ Perform the testing
- ♦ Measure the results.

An example of the above procedure can be found in the specification of the SPRING project that goes afterwards.

Besides, due to the increasing complexity of today's applications and the competitive pressures and costs of application failure, the demand for powerful **test management** is even greater. Since one has to handle with many projects, the best approach is adopting a good test management methodology that helps to centralize, organize, prioritize and document the tests to ensure coverage of the requirements, consistency and reusability of the tests campaigns.

A good test management methodology will help to track all the tests scripts generated in a department, to know at any moment what tests have been generated, what tests have been accomplished all right, what errors were found and if they have been solved…etc.

The test planning should start as soon as the project requirements have been stated. This parallelism with development ensures that the knowledge put into the design application is not lost in designing the testing strategy.

This is how both of the projects that will be explained below relate with each other. For the first project was necessary to generate an automated script which tested Microsoft Office Project Server 2003 interfaces (the web-access interface and the client interface). This part was made with a test tool for interfaces called WinRunner. Afterwards, another application called LoadRunner was used to launch the script on different machines at the same time, to simulate the load of multiple users working at the same time with Microsoft Office Project Server 2003. Afterwards, the resulting parameters were studied.
The second project was performed for a department that carried out with the regression of some of the software products developed by Amadeus. This implied hundreds and hundreds of tests that had to verify the software products every week to ensure that new modifications on this software did not affect to the functionalities that previously worked. In this department, it was urgent to improve their management methodology. Therefore, the project bases on a study about the possibility that TestDirector Management Tool from Mercury was capable to fulfil their requirements. For that, many customisations of the product were required with visual basic script programming.

Testing is gaining percentage of time into software development processes. Besides, it is becoming more and more important the achievement of error-free products and therefore many companies develop software with this goal. But it is also fundamental to find the right tool to do it because it can save a lot of time and as a consequence a lot of money. There is a web that assists people in selecting the right tool for testing: http://www.testtoolevaluation.com/

Regarding related work achieved I have evidence that in the company were I was working, my department was pioneer in customising the last version of TestDirector Management Tool and therefore, I had to present my work to other departments interested on that. Besides, Opodo, an online travel company owned by Amadeus, was interested in the customisation too, and we helped them out with it.
Both projects were achieved having a close relation with Mercury Support to ask for licenses and to get in contact with other people working in the same field. Each day, there are many companies looking forward to give more importance to their software testing processes and to do it efficiently. Regarding this, I have found many papers were professionals from different companies express their interest regarding tests methodologies. Here there are few of them:

"At Iron Mountain, our current mix of Mercury products includes over 500 scripts in Mercury WinRunner and Mercury LoadRunner. We use these to manage regression testing for our Oracle e-Business Suite, including financials, HR, iLearning, and property management modules. With an average of two Oracle upgrades a year, these scripts save us at least a month's manual regression test time."

"Punky McLemore is a quality assurance (QA) testing manager at Hewlett-Packard. Prior to working at HP, McLemore was a QA testing manager at a large U.S. financial institution, where she was responsible for testing all new applications for this multi-billion-dollar enterprise. The institution's QA staff employed up to 80 testing specialists and the number of Mercury TestDirector® users reached 120. McLemore has 16 years of experience in QA testing and has been a Mercury customer/user for seven years."

"In commercial software development organizations, increased complexity of products, shortened development cycles, and higher customer expectations of quality have placed a major responsibility on the areas of software debugging, testing, and verification. "**Brent Hailpern and Brent Hailpern** , *IBM Research Division*. "

4) Why Mercury products

Mercury is number one in testing products. But there are two other companies (IBM Rational and Segue) that together with Mercury share almost the whole testing software market. Amadeus have been working with Mercury for about six years already. Looking to the products of these three companies the features are very similar, only for very specific tests would be better to choose one instead of another.

The thesis will be performed with Mercury products due to it is the one that Amadeus purchased few years ago. The reason why Amadeus started using Mercury products was to test one of Amadeus products (Vista) which uses scriplet objects. At that moment, Mercury was the only company to develop special features to support this.

There are basically three Mercury products that were used in this thesis. TestDirector which is a Test Management tool, LoadRunner which allows server side load testing and WinRunner to test interfaces. And the three of them matched the project requirements.

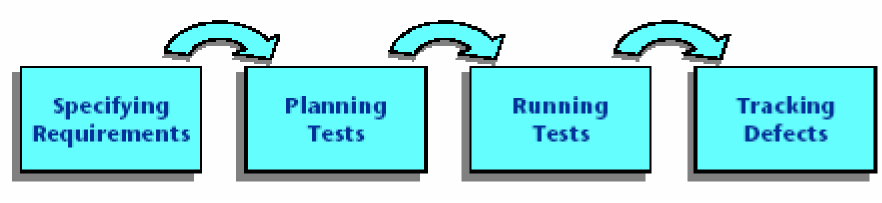
4.1) TestDirector

TestDirector is Mercury Interactive Web-based test management tool. TestDirector helps to organize and manage all phases of the application testing process, including specifying testing requirements, planning tests, executing tests, and tracking defects.

4.1.1 The TestDirector Testing Process

TestDirector offers an organized framework for testing applications before they are deployed. Since test LANs evolve with new or modified application requirements, you need a central data repository for organizing and managing the testing process. TestDirector guides you through the requirements specification, test planning, test execution, and defect tracking phases of the testing process.

The TestDirector testing process includes four phases:



Specifying Requirements
In this phase the tester should:

- Define testing scope: determine the test goals, objectives and strategies.
- Create requirements
- Detail requirements
- Analyze requirements specification: generate reports and graphs to assist in analyzing the testing requirements. Review the requirements to ensure they meet the testing scope defined.

Planning Tests
In this phase the tester should create a test plan based on the testing requirements. These are the steps to follow:

- Define testing strategy
- Define test subjects: divide **the application to test into subjects or functions to be tested**. Build a test plan tree to hierarchically divide the application into testing units or subjects.
- Define tests: determine the type of test it is needed for each subject.
- Create requirements coverage: link each test with a testing requirement.
- Design test steps: Develop manual tests by adding steps to the tests in the test plan tree and decide what tests should be automated
- Automate tests: create the corresponding tests scripts.
- Analyze test plan: generate reports and graphs and review the tests to determine their suitability to the testing goals.

Running Tests
After building a test plan tree, it is necessary to run the tests to locate defects and assess quality. In this phase the following tasks should be performed:

- Create test sets: determine which test to include in each test set
- Schedule runs: schedule tests execution and assign tests to testers
- Run tests
- Analyze test results: generate reports and graphs to analyze the results and to determine if there is any defect detected.

Tracking defects
Locating and repairing application defects efficiently is essential to the testing process. Defects can be detected and added **during all stages of the testing process**. In this phase, perform the following tasks:

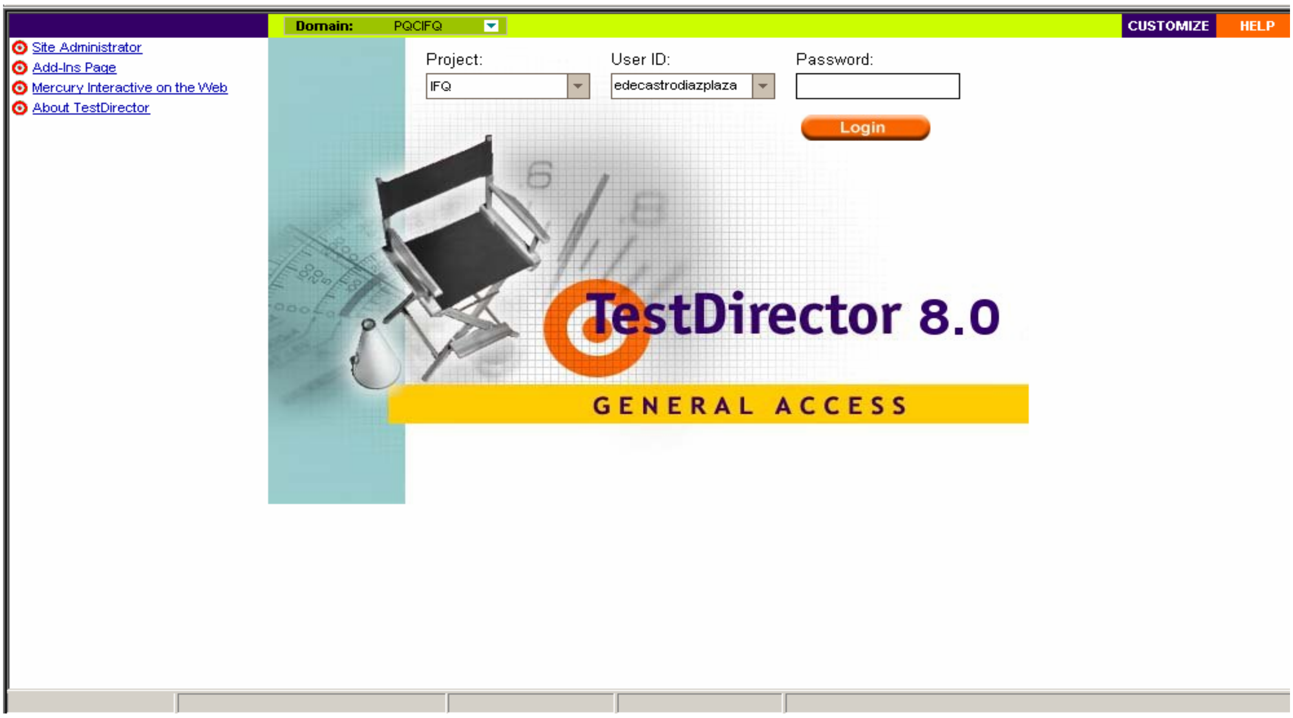
- Add defects: report the new defects detected
- Review new defects and determine which ones should be fixed
- Repair open defects
- Test new build: test the application until defects are repaired.
- Analyze defect data: generate graphs and reports to assist in analyzing the progress of defects repaired and to determine when to release the application.

4.1.2 TestDirector access

TestDirector does not need to be installed in the users local PC. TestDirector is installed in the company web server and it is accessible through web server.

The first time that TestDirector is run in a local computer, the application is downloaded. Then, each time TestDirector opens, it automatically carries out a version check. If TestDirector detects a newer version, it downloads the latest version to the local machine.

This makes it easier for the users to work with it and it is more flexible since it can be accessed from different operating systems or from home. Besides, new releases of the product will only need to be updated in the server machine.



Domain: PQCIFQ

Project: IFQ

User ID: edecastrodiazplaza

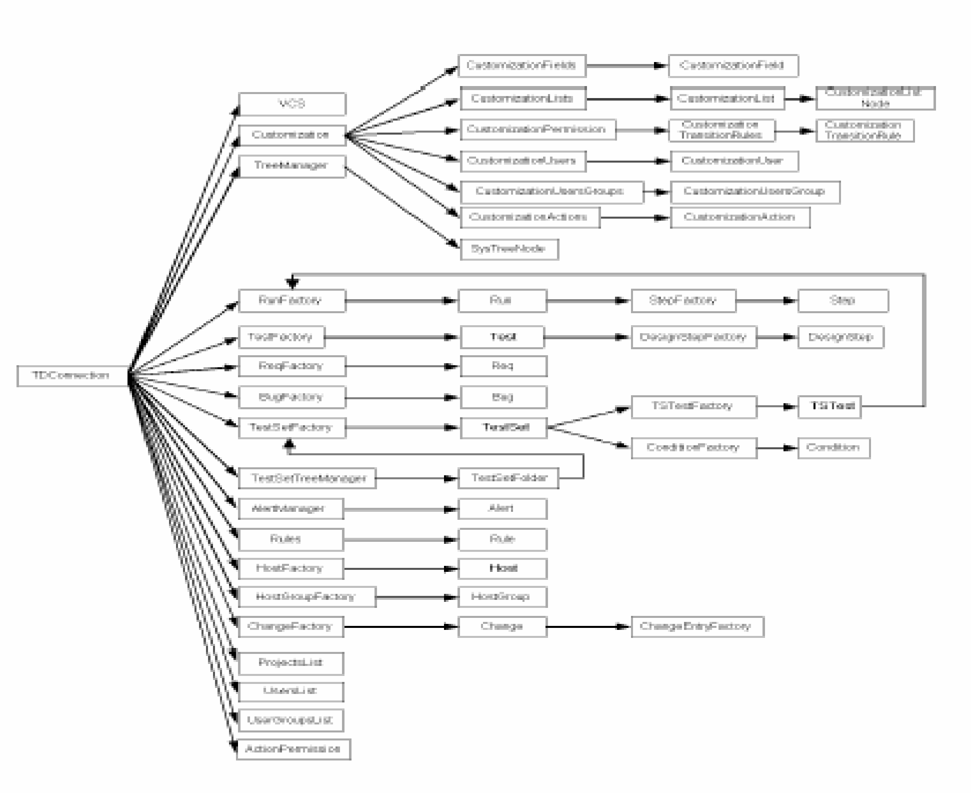
Password:

4.1.3 TestDirector customisation possibilities:

TestDirector gives the opportunity to customise any specific requisite. This is facilitated through OTA (Open Test Architecture) and the workflow.

4.1.3.1 The OTA:

Using TestDirector’s Open Test Architecture, it is possible to integrate specific requirements and configuration management tools, defect tracking tools, third-party and custom tools, and modeling applications. Execute tests remotely on multiple hosts across a network, and analyze the test results from within the TestDirector environment. Besides, TestDirector COM-based API enables an application to create, retrieve, and update TestDirector project data.



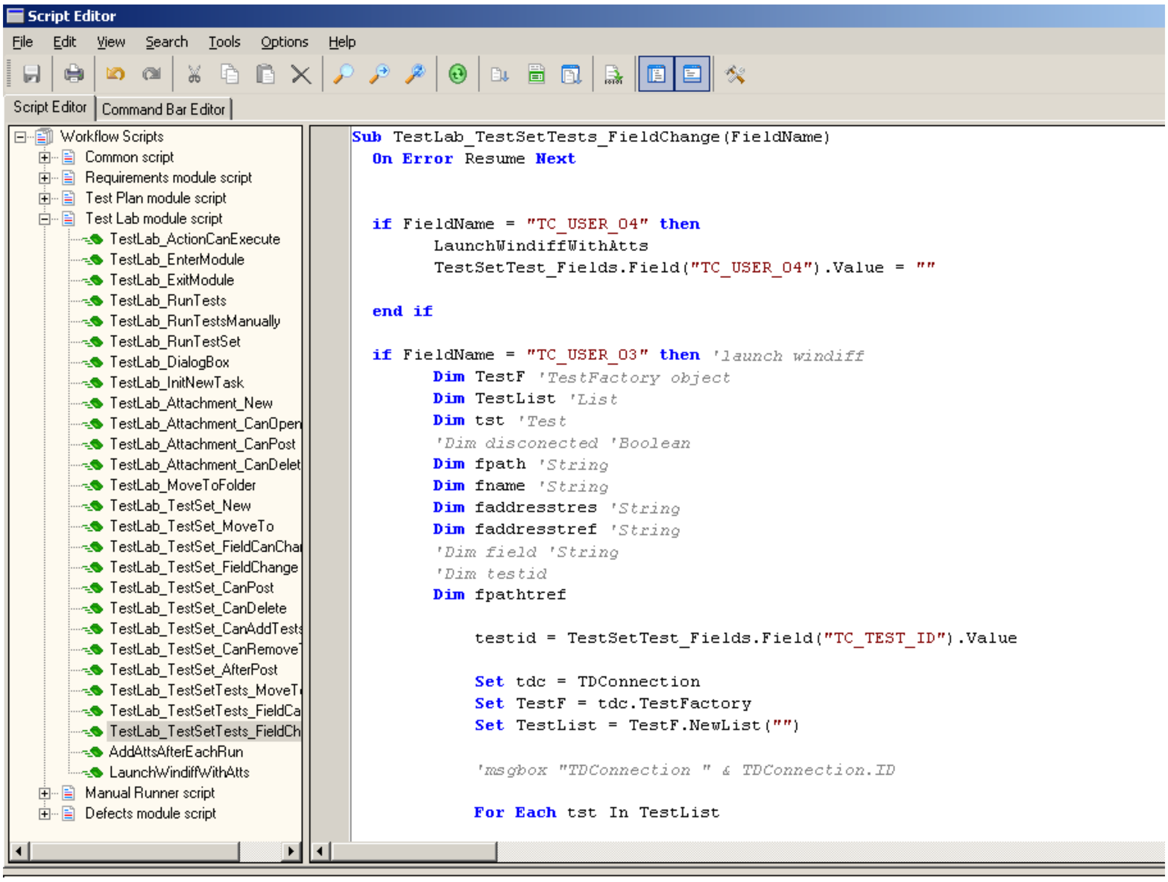
4.1.3.2 The workflow

As a TestDirector administrator, it is possible to create Visual Basic scripts that **control the workflow in a TestDirector project** and restrict and dynamically change the fields and values that are available in each TestDirector module.

For example, it is possible to create a button that launches windiff for the files attached to the selected test in TestLab. (How to do this is better explained in task number 1). Another example would be to send an email before the run of a test campaign is executed.

The script editor in TestDirector has different functions that will be executed in case that an event happens. For example: a button is clicked or a field changes its value. The administrator can write there a VBScript code that will be executed when that event happens.

There are also some context variables available in the workflow. These variables contain information about the object for which the current event works. This allows the programmer to know what are the value of the fields when the event happens and the function is called.

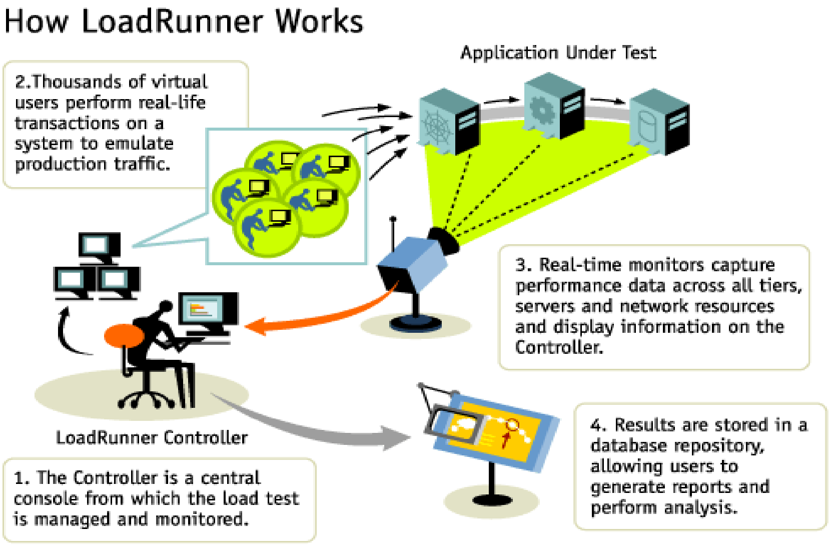


```
Sub TestLab_TestSetTests_FieldChange(FieldName)
  On Error Resume Next


  if FieldName = "TC_USER_04" then
        LaunchWindiffWithAtts
        TestSetTest_Fields.Field("TC_USER_04").Value = ""

  end if

  if FieldName = "TC_USER_03" then 'launch windiff
        Dim TestF 'TestFactory object
        Dim TestList 'List
        Dim tst 'Test
        'Dim disconected 'Boolean
        Dim fpath 'String
        Dim fname 'String
        Dim faddresstres 'String
        Dim faddresstref 'String
        'Dim field 'String
        'Dim testid
        Dim fpathtref

            testid = TestSetTest_Fields.Field("TC_TEST_ID").Value

            Set tdc = TDConnection
            Set TestF = tdc.TestFactory
            Set TestList = TestF.NewList("")

            'msgbox "TDConnection " & TDConnection.ID

            For Each tst In TestList
```

4.2) LoadRunner

LoadRunner is a load testing tool which allows server side load testing. It can connect to TestDirector, so scripts, scenarios and test campaigns results can be stored to assess application performance evolution.

The architecture of LoadRunner is a Windows server (called Controller), connected to “injector” machines also called “load generators” (see picture below)


How LoadRunner Works
Application Under Test
2.Thousands of virtual users perform real-life transactions on a system to emulate production traffic.
3. Real-time monitors capture performance data across all tiers, servers and network resources and display information on the Controller.
LoadRunner Controller
1. The Controller is a central console from which the load test is managed and monitored.
4. Results are stored in a database repository, allowing users to generate reports and perform analysis.

4.3) WinRunner

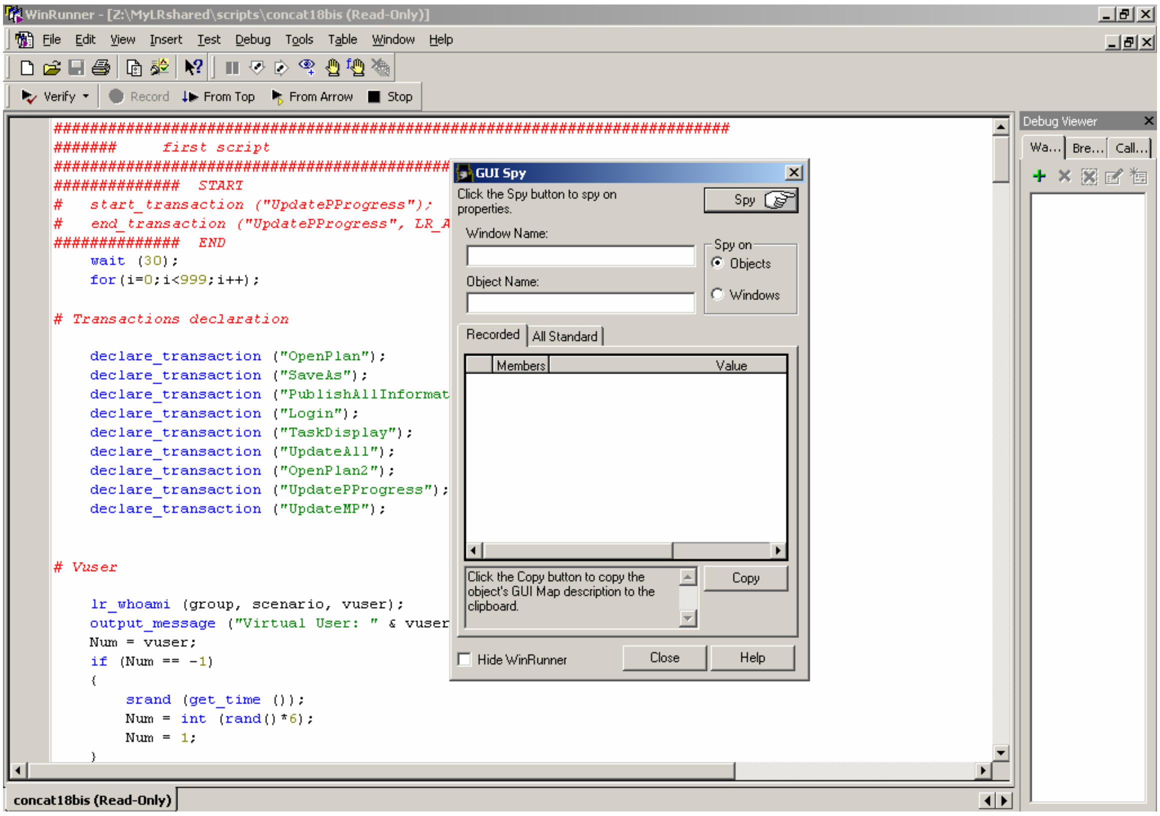
WinRunner is a functional testing tool for User Interface with strong integration in TestDirector. It can be used, as a “load generator” in combination with LoadRunner.

WinRunner helps to automate the testing process. It is meant to create adaptable and reusable test scripts that challenge the functionality of the applications.

WinRunner facilitates test creation by recording how the user works on the application. When the user points and clicks GUI (Graphical User Interface) objects in the application, WinRunner generates a test script in the C-like Test Script Language (TSL).

The difficulty in WinRunner is to ensure that the GUI map recognizes all the objects properly and to parameterize and synchronize the code in order to make it reusable and able to adapt to different speeds of events depending on the circumstances.

This is how WinRunner and the GUI map look like:



WinRunner - [Z:\MyLRshared\scripts\concat18bis (Read-Only)]

```
###########################################################################
#######     first script
############################################
##############  START
#   start_transaction ("UpdatePProgress");
#   end_transaction ("UpdatePProgress", LR_A
##############  END
    wait (30);
    for(i=0;i<999;i++);

# Transactions declaration

    declare_transaction ("OpenPlan");
    declare_transaction ("SaveAs");
    declare_transaction ("PublishAllInformat
    declare_transaction ("Login");
    declare_transaction ("TaskDisplay");
    declare_transaction ("UpdateAll");
    declare_transaction ("OpenPlan2");
    declare_transaction ("UpdatePProgress");
    declare_transaction ("UpdateMP");


# Vuser

    lr_whoami (group, scenario, vuser);
    output_message ("Virtual User: " & vuser
    Num = vuser;
    if (Num == -1)
    {
        srand (get_time ());
        Num = int (rand()*6);
        Num = 1;
    }
```

GUI Spy

Click the Spy button to spy on properties.

Window Name:

Object Name:

Spy on: ● Objects ○ Windows

Click the Copy button to copy the object's GUI Map description to the clipboard.

□ Hide WinRunner

concat18bis (Read-Only)

5) Spring project

5.1) Introduction

Amadeus' top management requires a better management of the project portfolio of the company. Amongst practical actions that can be endeavored, a more thorough management of the human resources is a key for success as it would allow clear visibility on how many and when the human resources are allocated to projects.

Today the data used to manage work and resources is spread amongst a number of software tools that harshly communicate with each other.

Microsoft Project Server 2003 (MSP 2003) provides planning and resource management under a collaboration model. An evaluation has shown that the tool has reached an acceptable level of functionality and capacity, to cope with Amadeus requirements.

All these topics are addressed by the SPRING project.

The SPRING project will investigate the performance, reliability and scalability of MSP 2003 in the following context:

- scenarios exercising “time reporting” and “project plan management” activities
- two different client access types: from LAN workstations and from a Terminal server machine
- production infrastructure (application and database servers) hosted in Nice

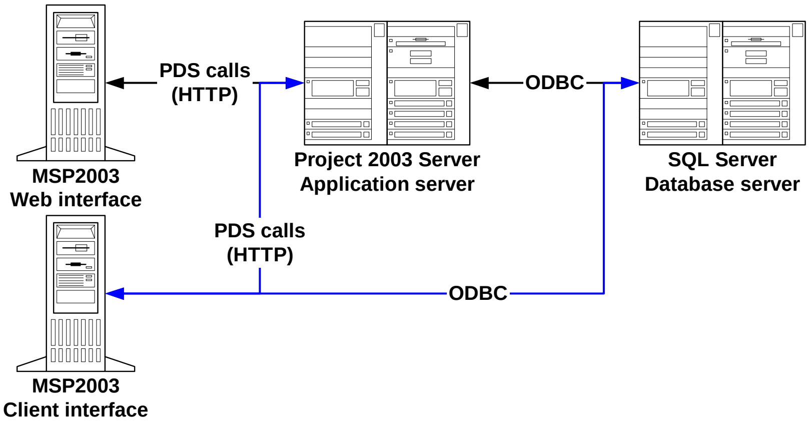
Thanks to the infrastructure dedicated to the tests, reproducible tests have been made possible and workloads are exhibiting a very low variance from one run to another.

The following chapters describe the aspects that have been covered by the testing which gives evidence about that there is no risk in terms of performance and capacity by deploying the SPRING project. However, a database management problem was detected due to the higher load, resulting in a greatly increased response time.

5.2) Testing goals

MSP 2003 deployment in Amadeus is part of the SPRING project; therefore the responsiveness and scalability of MSP 2003, on the selected production platform, must be verified.

MSP 2003 is a three-tiered application, as depicted below:



Client tiers:

- Microsoft Project Web Access 2003 (web interface)
- Microsoft Office Project Professional 2003 (client interface)

Application tier:

- Microsoft Office Project Server 2003 (application server)

Database tier:

- SQL server 2003 (database server)

The objectives of the performance tests may be split into several parts:

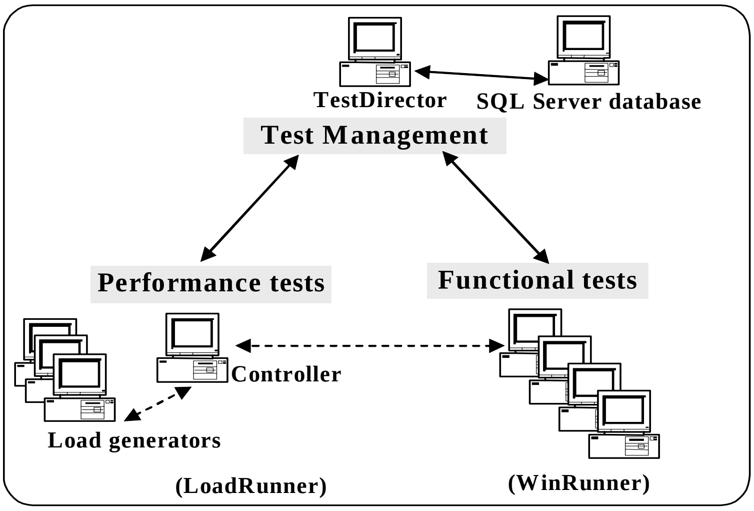
- How does the system respond to great number of users using MSP Web Access?
- How does the system respond to great number of users using the MSP client?
- How does the system respond to a peak activity?
- How does the system behave over time under a steady traffic?

In any cases we would measure:

- The response time of the system, from a end-user perspective
- The system resources consumption on both MSP Server and the SQL Server

5.3) Testing software integration:

The picture below shows the tools integration:



In the beginning, it was planned to use WinRunner to simulate the end-user interaction with MSP2003 client and LoadRunner for MSP2003 Web access part. However, the data obtained by LoadRunner when recording the interactions with Project2003 server was completely illegible. These data came in cipher, so it was impossible to understand it and parameterize the script in order to automate it for different users.
Consequently, all scripts were recorded with WinRunner but LoadRunner controller tool was required as well to launch them and coordinate in several computers.

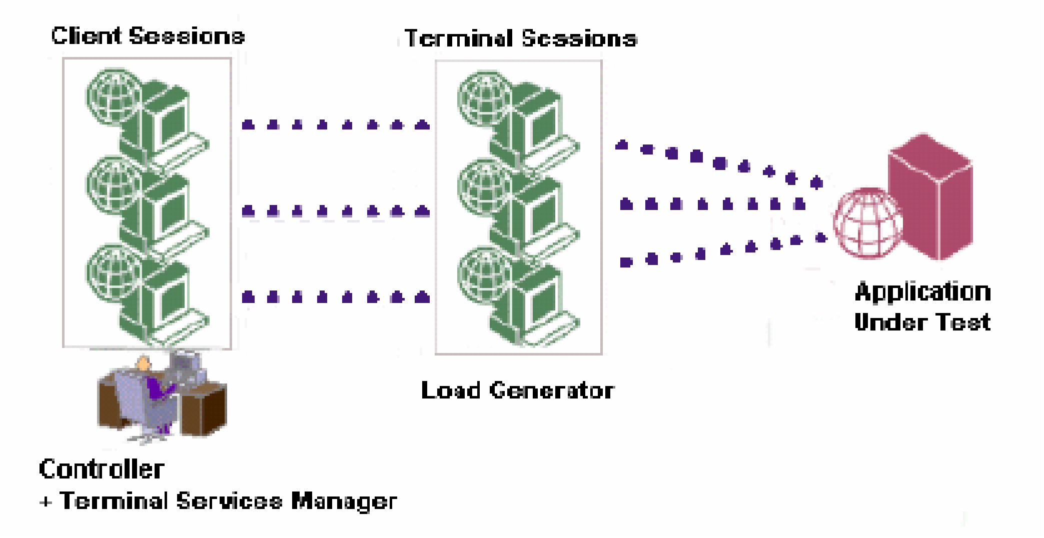
The number of users that execute the scripts was limited to five which is the number of LoadRunner licenses for running concurrent WinRunner scripts. This was a bit of disappointment, since LoadRunner license allows for running up to 350 concurrent LoadRunner scripts.

Nevertheless it was the first time that WinRunner scripts were used as GUI Virtual users in LoadRunner: this is a nice feature when the client software must be included in the test. It allows validating also the client performance and robustness.

5.4) Test Bed

The test platform must allow for two different testing configurations:

- The first one with the users connected on the LAN
- The second one for Amadeus users accessing the application from a remote location (i.e. MIAMI). To reduce the bandwidth usage on the WAN, users will connect through Terminal Services (see figure below)



In a case, WinRunner scripts are executed concurrently, either on separate workstations or in distinct sessions on the Terminal Server machine.
Running WinRunner in a Terminal Service session requires the installation of a license server and specific settings on the Controller (Annex A).

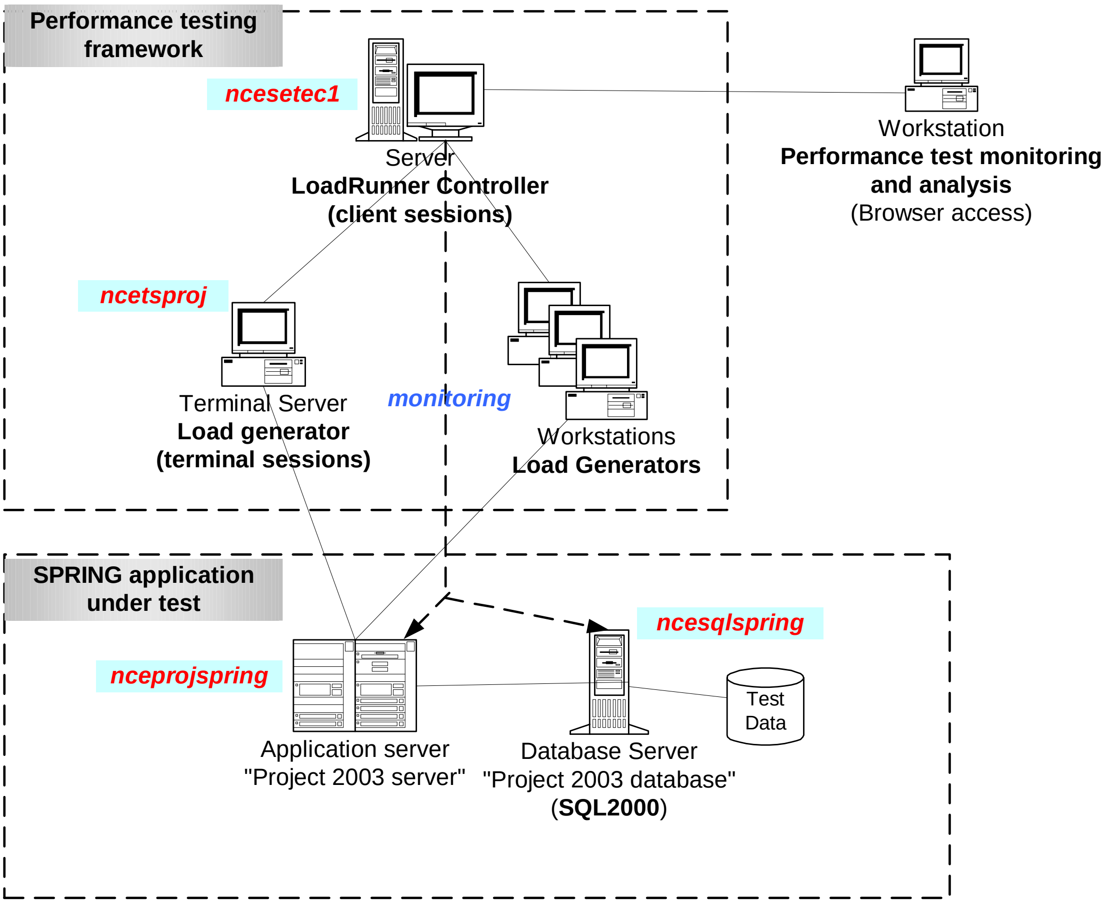
All tests have been run on the SPRING production infrastructure during dedicated time slots where all traffic was stopped: SPRING pilot users were prevented from using MSP2003 (no access granted).

A production-like test database is used: 50 plans with about 500 tasks and 30 resources assigned.

Before each test sessions, we:

- backup the database prior testing
- restrict access to “test” users only
- run the tests
- Restore the database after testing
- Restore access to SPRING pilot users

The full test platform is depicted below:



5.5) Test script

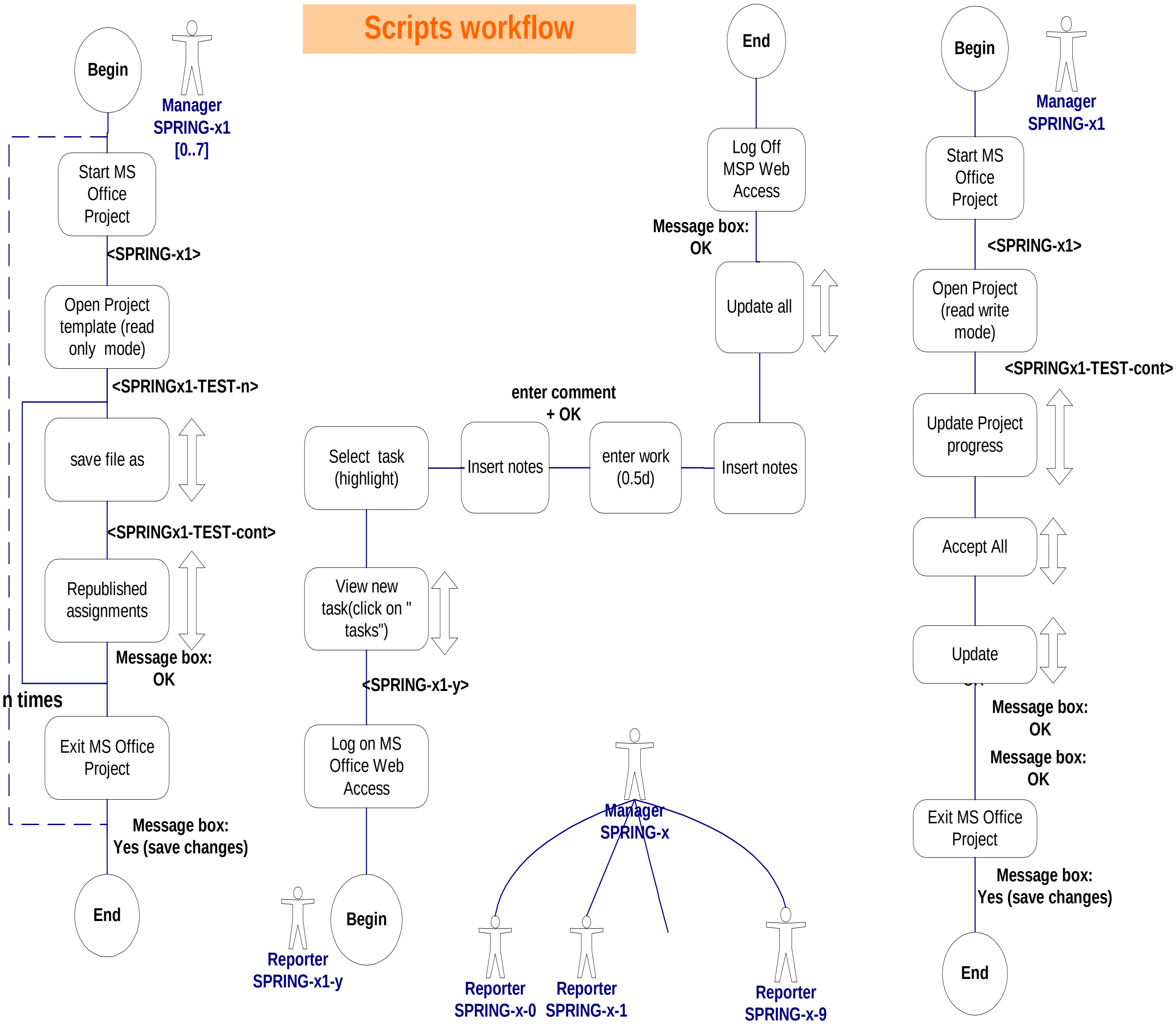
5.5.1) Business process

The script will reproduce the way in which Amadeus is going to use MSP2003. Therefore, scripts will simulate the work that the managers and reporters will perform.

A manager can create and follow-up several projects and he will assign tasks to team members. These team members (also called “reporters”) will report the time spent on each tasks, on a daily or weekly basis.

Consequently, there are three functional stages defined in the automated script:

- The manager creates a plan by opening a project template and saving it with a different name. Afterwards, the plan is republished and Microsoft Office Project is closed.
- The reporters that got the tasks assigned, log into Microsoft Project Web Access 2003 and insert notes and report time on the tasks, update all the changes and then log off.



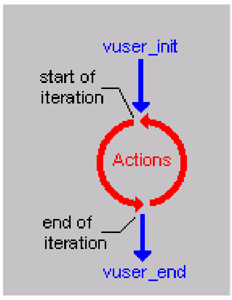
- Once that all the reporters have completed their tasks, the manager edit the project plan (read/write mode), accept and update all the changes made by the reporters. Then Microsoft Project is closed.

They are depicted more precisely below:

This script is implemented as a WinRunner script. It must be reproducible, since several iterations of the scripts must be performed during a test.

5.5.2) Script design

The structure of a script is the following:



Init section will establish the socket connection with the server and perform the travel agent login.

The set of actions must let the ACE application in such a state that a new iteration can be performed

End section will close the main menu window and perform a logout.

To generate the script in WinRunner, the first step is to record what the script should do. Then, it is necessary to **parameterize** it so that it can work for different users. It has also to be **synchronized** due to depending on the workload, the events will happen sooner or later: this is the trickiest part.

Events have to be waiting for the previous ones and sometimes it is not easy to synchronize them. It is possible to be waiting for a window to appear, or for a button in an object or for a text on window frame.

Sometimes, there are windows that appear depending on the server circumstances so it is necessary to take this into account and wait for it only in the case that it appears in a limited time out.

Then, the test has to be designed so that it is reproducible and repeatable, so the state at the beginning and at the end of the script should be the same.

5.6) Measurements

For each scenario, we are measuring:

- Application responsiveness (end-to-end) from the WinRunner script
- System resource usage on the different application tiers (application and database servers)

Transaction name	**Transaction description**
Login	**Time to login from Web Access logon page**
OpenPlan	**Time to open the project template plan.**
OpenPlan2	**Time to re-open the plan**
PublishAllInformation	**Time to publish the new plan information (⚠ it is a background task)**
SaveAs	**Time to save the project template as a new plan**
TaskDisplay	**Time to display the task assigned to the "Reporter" (Web Access)**
UpdateAll	**Time to update the task assigned to the "Reporter", after changes (Web Access)**
UpdateMP	**Time to Update, after acceptation, the plan with the "reporters" updates**
UpdatePProgress	**Time to get the "Progress" from the plan updated by the "reporters"**

5.6.1) Performance

These are pure performance figures, obtained from end-user perspective for one user workload. There is no other traffic flowing to the application and database server.

All Web interface transaction response times are below 2 sec.
All Client interface transaction response times are below 10 sec, except for “Update” (Accepted task changes in Project) which is 20 sec.

It must be noticed that “Publish All information” transaction cannot be measured, since it runs in background on the server.

However, significant and fast performance degradation was detected for the “Update” transaction (up to 60 sec.), during long runs. **This could be a database server tuning issue, so it must be carefully investigated by specialists**.

5.6.2) Scalability

Since we had to run WinRunner scripts as GUI virtual users in LoadRunner, we were limited to five virtual users (license limitation).

In fact we didn’t manage to run more than three virtual users concurrently without application fault or script error (due to application behavior change).
However, three virtual users were running without think time between the transactions, so they generate a load equivalent to a much higher number of real users.

The table below shows the highest system resource consumptions we could reach for three virtual users:

Application Tier	**Total user transactions per minute**	**%CPU utilization Average**	**%CPU utilization Max**
Application server	**15**	**10**	**50**
Database server		**25**	**80**
Terminal server		**5**	**40**

We didn’t manage to reach 100% CPU utilization on any Tier of the application.

We must put these figures side by side with the **production traffic which is extrapolated to be one transaction per minute**, based on the following assumptions:

- 450 reporters reporting time, via the Web interface, once every week
- 160 managers working with the client interface to:
    - update up to five project plans once a week
    - update/accept progresses of ten projects once a week

So, assuming that the transaction mix of our scripts represents approximately the production traffic, the capacity of the system is very much sufficient to cope with the estimated traffic. The bottleneck, if any, will likely be the database server.

Nevertheless, we noticed that two transactions are seriously affected by the database log growth: “Open” and “Update” projects. Thus, as the scenario creates new projects in the database, the response times increases rapidly: from 10 sec. to 70 sec. (for an “open” project transaction). **This behaviour must be carefully analyzed and explained**.

5.6.3) Robustness

Microsoft Project Server 2003 appeared to be robust: it never crashed during the tests and no system resource leaks were detected.

However, application errors happened in a situation of transaction concurrency: hot fixes has been provided by Microsoft support that solved the problems (two times).

Fair enough, these application errors never lead to data corruption. Nevertheless we believe that concurrent access to MS plan (from the MSP client interface) is not well managed for a high level of transaction concurrency. This is probably a consequence of MSP 2003 being a reengineering and adaptation of the original MSP standalone version rather than a full redesign.

In any case, this not seen this as a serious problem in the context of SPRING: there will be few people accessing the plans (managers) and modifying it should not happen very frequently (compared to the transaction duration).

5.7) Test results

The tests didn’t run flawlessly.

The main problems arising were:

- The synchronization between the different steps of the script has to be reworked
- Errors/Exceptions messages have been raised by MS Project. They have been addressed to Microsoft support which provided hot-fixes
- We were not able to successfully run the script with more than three concurrent virtual users, the limitation coming from the application behaviour rather than from the system resources.

All scenarios have been run several times to assess reproducibility of the results. Therefore only selected Analysis reports are presented in the following sub-chapters.

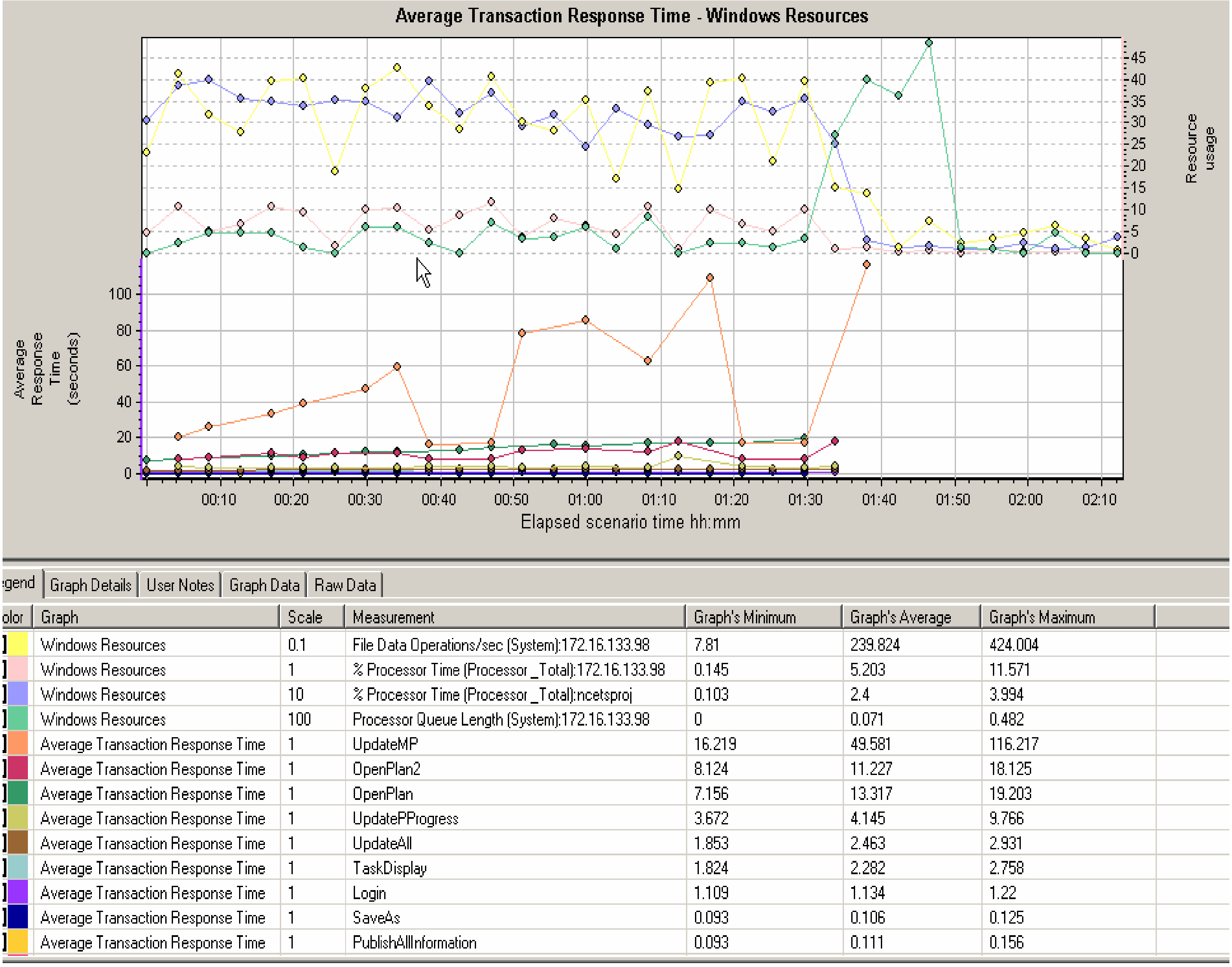
5.7.1) Scenario with 1 Virtual user

This scenario generates a constant load on the application. However, the size of the database growths since a new plan is created for each script iteration.

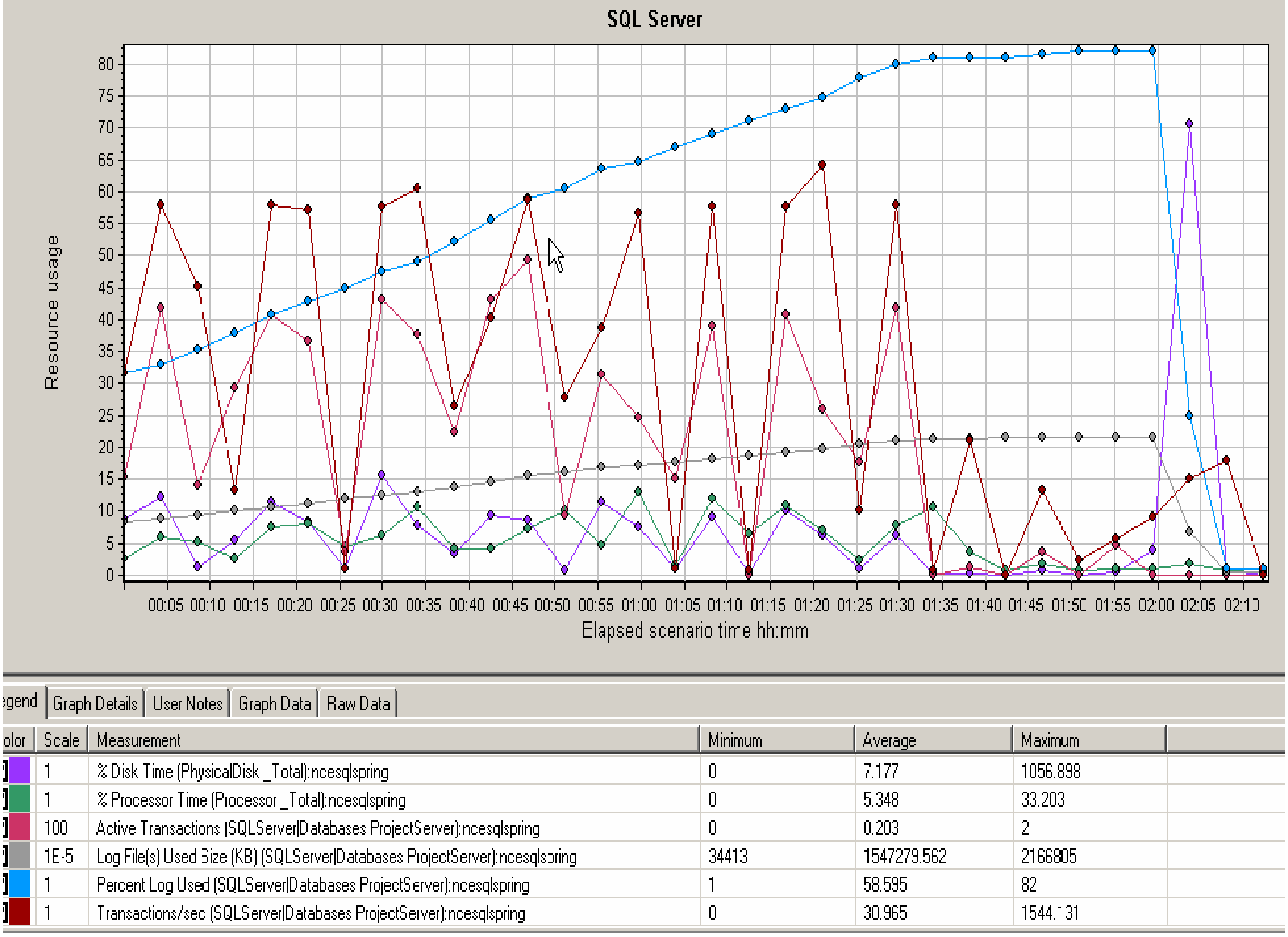
The graphs below shows an abnormal response time increase for the transaction “UpdateMP”: it increases with the number of script iterations, so with the number of plan created in the database.

The Windows Resources graphs doesn’t show any increase of application server resources, when SQL server graphs shows a log file size constant increase (blue line).

The system resource consumption is mainly on SQL server machine: a ratio of Two (2) between application and database server.



olor	Graph	Scale	Measurement	Graph's Minimum	Graph's Average	Graph's Maximum
🟨	Windows Resources	0.1	File Data Operations/sec (System):172.16.133.98	7.81	239.824	424.004
🟥	Windows Resources	1	% Processor Time (Processor _Total):172.16.133.98	0.145	5.203	11.571
🟦	Windows Resources	10	% Processor Time (Processor _Total):ncetsproj	0.103	2.4	3.994
🟩	Windows Resources	100	Processor Queue Length (System):172.16.133.98	0	0.071	0.482
🟧	Average Transaction Response Time	1	UpdateMP	16.219	49.581	116.217
🟥	Average Transaction Response Time	1	OpenPlan2	8.124	11.227	18.125
🟩	Average Transaction Response Time	1	OpenPlan	7.156	13.317	19.203
🟨	Average Transaction Response Time	1	UpdatePProgress	3.672	4.145	9.766
🟫	Average Transaction Response Time	1	UpdateAll	1.853	2.463	2.931
🟦	Average Transaction Response Time	1	TaskDisplay	1.824	2.282	2.758
🟪	Average Transaction Response Time	1	Login	1.109	1.134	1.22
🟦	Average Transaction Response Time	1	SaveAs	0.093	0.106	0.125
🟨	Average Transaction Response Time	1	PublishAllInformation	0.093	0.111	0.156



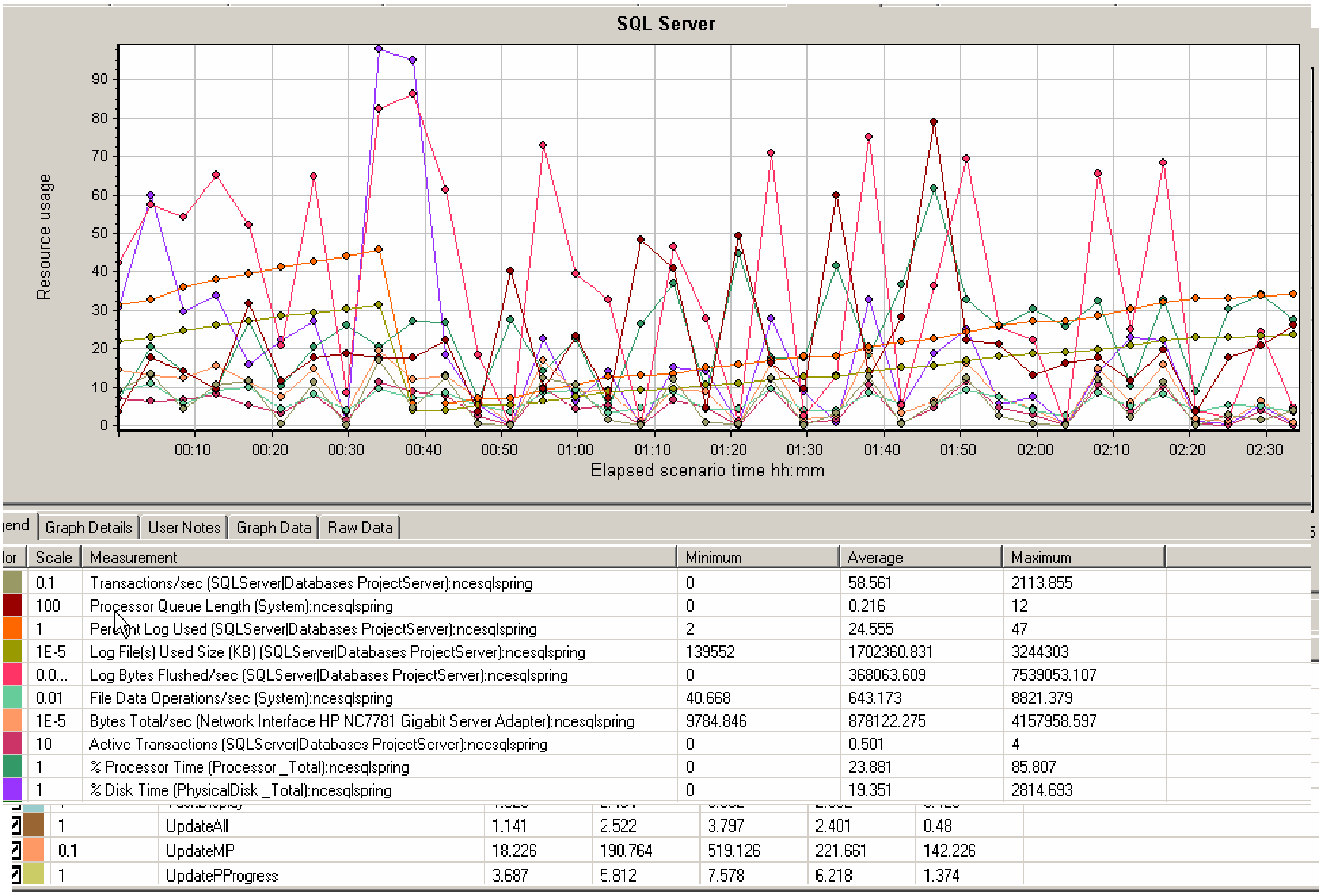
olor	Scale	Measurement	Minimum	Average	Maximum
🟪	1	% Disk Time (PhysicalDisk _Total):ncesqlspring	0	7.177	1056.898
🟩	1	% Processor Time (Processor _Total):ncesqlspring	0	5.348	33.203
🟥	100	Active Transactions (SQLServer:Databases ProjectServer):ncesqlspring	0	0.203	2
⬜	1E-5	Log File(s) Used Size (KB) (SQLServer:Databases ProjectServer):ncesqlspring	34413	1547279.562	2166805
🟦	1	Percent Log Used (SQLServer:Databases ProjectServer):ncesqlspring	1	58.595	82
🟥	1	Transactions/sec (SQLServer:Databases ProjectServer):ncesqlspring	0	30.965	1544.131

5.7.2) Scenario with 3 Virtual users

This scenario brings out evidence that there is something wrong with the database management: the higher load shows a drastic response time increase for the transactions opening a plan (OpenPlan, OpenPlan2) and UpdateMP.

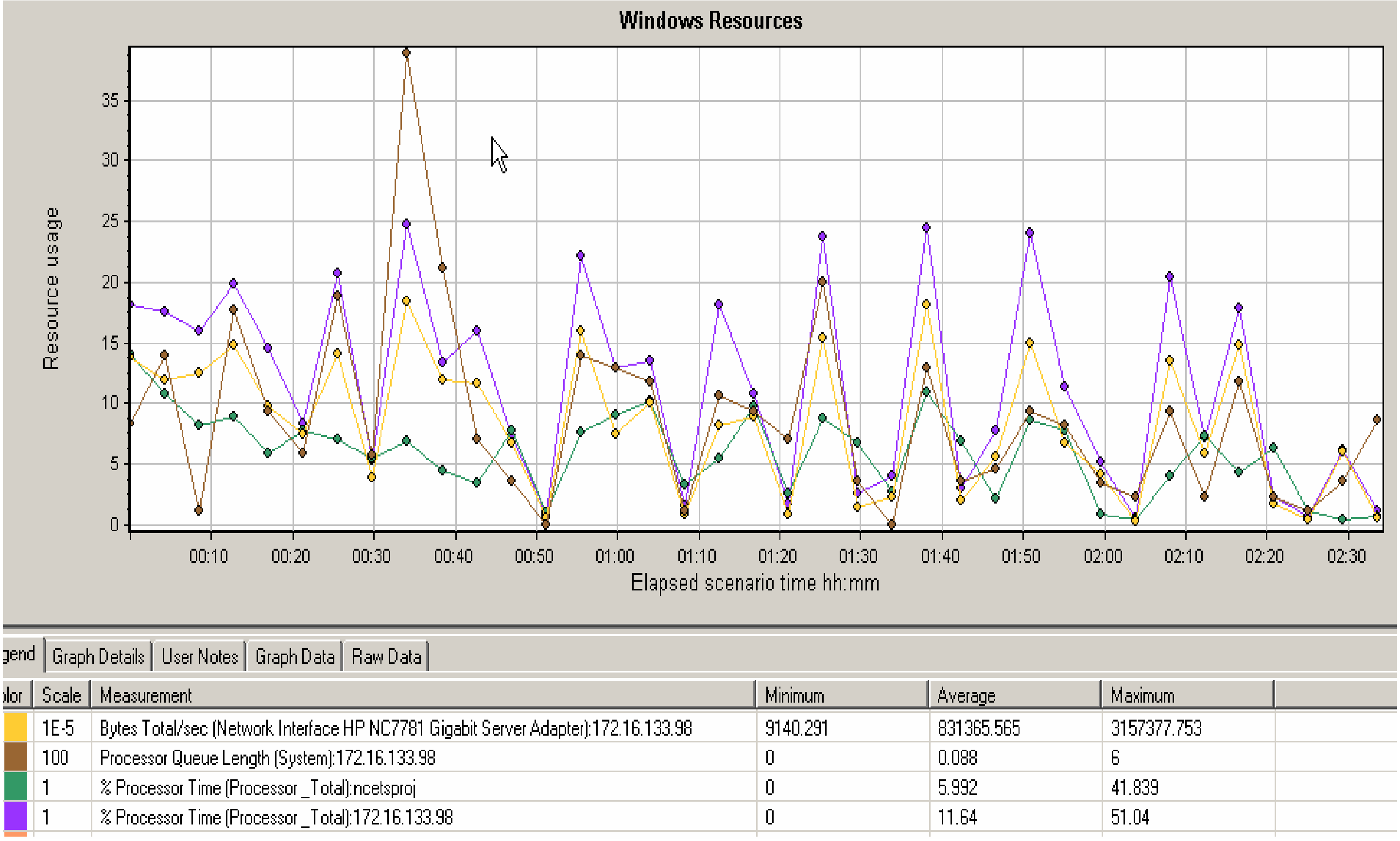
Like for the previous scenario, the response time "recovers" at 01:00:00 and 01:30:00, then it increases again.
Strange enough, it doesn't correspond to a database check-point (00:35:00) when a lot of disk access happened.

Repeating the scenario will show exactly the same graphs shape.



lor	Scale	Measurement	Minimum	Average	Maximum
■	0.1	Transactions/sec (SQLServer|Databases ProjectServer):ncesqlspring	0	58.561	2113.855
■	100	Processor Queue Length (System):ncesqlspring	0	0.216	12
■	1	Percent Log Used (SQLServer|Databases ProjectServer):ncesqlspring	2	24.555	47
■	1E-5	Log File(s) Used Size (KB) (SQLServer|Databases ProjectServer):ncesqlspring	139552	1702360.831	3244303
■	0.0...	Log Bytes Flushed/sec (SQLServer|Databases ProjectServer):ncesqlspring	0	368063.609	7539053.107
■	0.01	File Data Operations/sec (System):ncesqlspring	40.668	643.173	8821.379
■	1E-5	Bytes Total/sec (Network Interface HP NC7781 Gigabit Server Adapter):ncesqlspring	9784.846	878122.275	4157958.597
■	10	Active Transactions (SQLServer|Databases ProjectServer):ncesqlspring	0	0.501	4
■	1	% Processor Time (Processor _Total):ncesqlspring	0	23.881	85.807
■	1	% Disk Time (PhysicalDisk _Total):ncesqlspring	0	19.351	2814.693

■	1	UpdateAll	1.141	2.522	3.797	2.401	0.48
■	0.1	UpdateMP	18.226	190.764	519.126	221.661	142.226
■	1	UpdatePProgress	3.687	5.812	7.578	6.218	1.374



lor	Scale	Measurement	Minimum	Average	Maximum
🟨	1E-5	Bytes Total/sec (Network Interface HP NC7781 Gigabit Server Adapter):172.16.133.98	9140.291	831365.565	3157377.753
🟫	100	Processor Queue Length (System):172.16.133.98	0	0.088	6
🟩	1	% Processor Time (Processor _Total):ncetsproj	0	5.992	41.839
🟪	1	% Processor Time (Processor _Total):172.16.133.98	0	11.64	51.04

5.8) Conclusion:

The study reveals that there is no risk in terms of performance and capacity deploying the SPRING project. However there are some items worthy of further testing and analyzing to avoid performance degradation with the database growth.

6) TestDirector migration, feasibility study and customisation

6.1) Introduction

This second project is about finding a solution to the need of updating the test management platform in the Internet and Front Office Quality team (IFQ team) in Amadeus. The previous used platform is Lotus Notes which is more a database than a test management platform and the specific features that had been implemented on Lotus Notes are not easily upgraded. This section will perform a feasibility study of the migration to a new test management platform having into account the features in the old platform and the team requirements in order to customise it to the team needs. It will involve understanding of the functionalities required, self-learning of the new platform, contacting other engineers developing similar jobs, and discussions with the technical support to ask for bug's patches and for technical details.
Test Director is the test management tool selected because it is highly customisable which allows to be well adjusted to the specific needs but on the other hand, a customisation is required in order to be able to get benefits of the complexity of the product. The last version of the product includes many new customisation possibilities which has made that many companies go for this product but however, all the customisation possibilities have not been explored yet and therefore all the IFQ needs had to be studied in order to see if the migration was possible.

In the following chapters, the thesis describes the specification of the migration, the steps followed in order to achieve them, alternatives solutions and why it was done in one way instead of another.

6.2) Migration specification

TestDirector is a test management tool and therefore it facilitates common test management features but there are some specific ones that IFQ team needs for their concrete work. This was why in the beginning there was some incertitude about the possibility of the migration. Here are the requirements that needed to be filled in order to make feasible the migration to TestDirector. Later on it will be explained how they were achieved.

Step	Bypass possible	Mandatory	Priority
1	Button in the 'Test Lab' section to call Windiff **Bypass:** Not applicable	Yes	1
2	Retrieve the focused script in the 'Test Lab' section. **Bypass:** Not applicable	Yes	1
3	Call an external tool in the 'Test Plan – Test Script' section for a VAPI-XP test script **Bypass:** Not applicable	Yes	1
4	Find a specific file within the attachments of a VAPI-XP test script **Bypass:** Not applicable	No	2
5	Pass some results of a script to the general report. **Bypass:** Possibility to set manually the general report	No	3
6	Re-use some parts of VAPI-XP scripts code (generic code, templates, dll…). **Bypass:** Not applicable	Yes	1

7	Automatically attach some result files in the attachment part of the ‘run test’ in the ‘Test Lab’ **Bypass:** Not applicable	No	1
8	Create a general report with all the mandatory data **Bypass:** Possibility to manage the general report manually and send to IFQ.	No	2
9	Generate a Web page with the same data present in the general report **Bypass:** Possibility to create manually the Web page	No	3
10	Send the general report by mail **Bypass:** Manual action possible	No	3
11	Verify if it is necessary to have SourceSafe in a local PC to extract a file in SourceSafe **Bypass:** Depends on task number 4.	No	3

6.3) Solutions

This part is really important because it shows the status of the different studies performed, how they were achieved, what problems I met, other possible ways to achieve it and the explanation about why I did it one way instead of another.

1. **Add button in the ‘Test Lab’ section to call Windiff**

Status: Done

Performance: In the TestLab part of the workflow in TestDirector, a new function that call Windiff tool with the reference and result files that are attached to the script has been implemented. So, it is possible to call it after any of the events that TestDirector handles. Now, it is called when a user click in the “launch Windiff with atts” button in the TestLab.

Alternatives solutions:

a) One possibility was to get the path of the files in the TestDirector server machine and launch windiff with those paths. However, this could bring future problems since it is supposed that all applications should communicate through the OTA instead of directly with the server machine itself. For this reason, the files are downloaded to the local machine and afterwards, windiff is called with the new paths in the local machine.

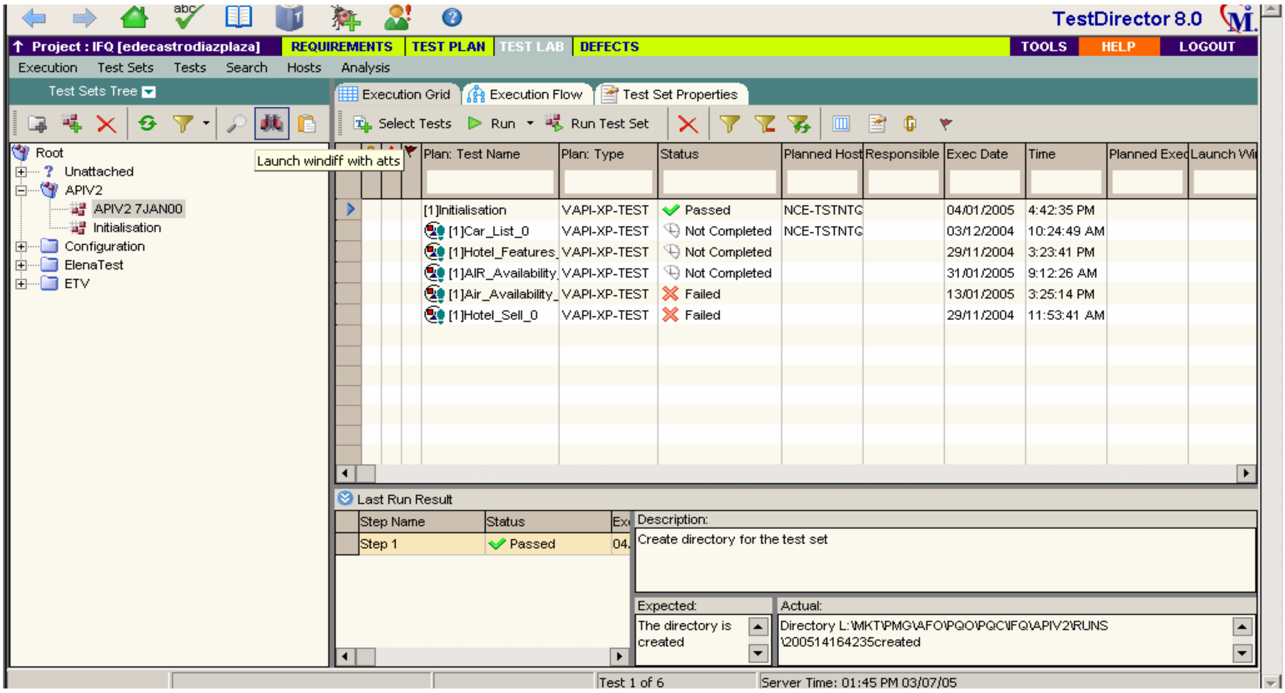
b) It is important to mention that there is a bug in the function that is made to download the attachments, and the bug consist in that the function ignores the location specified and downloads the file into a temporal folder:

```
att.Load True, "C:\"
```

So, to call windiff with the attachment it is fine if the files are in a temporary folder, and this function has been used. However, to download the attachments into a specific folder, it is necessary to use this work around:

```
Set ext = att.AttachmentStorage
filename = att.DirectLink
ext.Load filename, True
Set fso = CreateObject("Scripting.FileSystemObject")
'File = name with which the file is stored in the specified location
fso.CopyFile ext.ClientPath & "\" & filename, "D:\Documents\" & File
MsgBox "The file has been downloaded successfully"
```

msgbox att.FileName



Plan: Test Name	Plan: Type	Status	Planned Host	Responsible	Exec Date	Time	Planned Exec	Launch Wi
[1]Initialisation	VAPI-XP-TEST	✓ Passed	NCE-TSTNTG		04/01/2005	4:42:35 PM		
[1]Car_List_0	VAPI-XP-TEST	Not Completed	NCE-TSTNTG		03/12/2004	10:24:49 AM		
[1]Hotel_Features_	VAPI-XP-TEST	Not Completed			29/11/2004	3:23:41 PM		
[1]AIR_Availability_	VAPI-XP-TEST	Not Completed			31/01/2005	9:12:26 AM		
[1]Air_Availability_	VAPI-XP-TEST	✗ Failed			13/01/2005	3:25:14 PM		
[1]Hotel_Sell_0	VAPI-XP-TEST	✗ Failed			29/11/2004	11:53:41 AM		

Step Name	Status	Ex
Step 1	✓ Passed	04.

Description:
Create directory for the test set

Expected:
The directory is created

Actual:
Directory L:\MKT\PMG\AFO\PQO\PQC\IFQ\APIV2\RUNS \200514164235created

Test 1 of 6

Server Time: 01:45 PM 03/07/05

2. Retrieve the focused script in the ‘Test Lab’ section

Status: Done. This point is included in the previous part
Performance: In the workflow of the customisation part of TestDirector, there are some variables that allow the user to access to TestDirector fields.

Use {Object}_Fields(“{Field_Name}”) to access field by name.
Use loop on {Object}_Fields.FieldById(i) to access all fields in the collection.

Both methods allow working with the fields of the “current object.” The current object can be defined in the following way:
Current object type: The object type for which the current event works. Almost each event points to the object type for which the fields can be retrieved. For example, in Defects_Bug_... events, only Bug_Fields can be retrieved; in TestPlan_DesignStep_... events, only DesignStep_Fields can be retrieved.
Focused item: From all objects of the collection defined by event, only the fields of the currently focused object can be retrieved. For example, the test on which the cursor is placed or the current run in manual runner.
To retrieve the fields of other objects of the same/other object type, use TD API (OTA).
The first statement *{Object}_Fields(“{Field_Name}”)* can be used to retrieve any particular field by name.
The second statement *{Object}_Fields.FieldById(i)* is needed to go over all fields of the current object. For example, to reset the fields order

3. Call an external tool in the ‘Test Plan – Test Script’ part for a VAPI-XP test script.

Status: Done.
Performance: From VAPI-XP test scripts is possible to call another tools.

Solution:

a) res = XTools.run("windiff.exe", parameter, -1, TRUE)

b) Set WshShell = CreateObject("Wscript.Shell")

intReturn = WshShell.Run("windiff " & parameter, 1, True)

Note: The first solution seems to be better for the VAPI-XP since it is a method included in the SRunner library that is directly added by TestDirector for these tests. The other solution is needed when programming in the workflow of TestDirector

4. **Find a specific file within the attachments of a VAPI-XP test script**

The objective of this part is to find a file ".scr" within the attachments of a script. To solve it, a template script has been implemented. It checks if the file that has called it, has a concrete .scr file in the attachments.

5. **Pass some results of a script to the general report. For example adding into the general report the versions of different components when these ones are verified**.

Status: Done.

Performance: Four extra custom fields have been added, one for each product version. So, now the TestSet properties include as well APIV2 proxy version, APIV2 gateway version, APIV2 CornCore version and FareQuote database version. Once in the APIV2-XP tests the values for these variables are obtained, it is just necessary to set that value to the corresponding TestSet property in TestDirector.

For example:

```
        Var1 = "APIV2 proxy version: 4"
  CurrentTestSet.Field("CY_USER_06")= Var1
  CurrentTestSet.Post()
```

This code should be included in the Test_Main function of the VAPI-XP. It is important to know the following data:

Field Label	Field Identification
APIV2 proxy version	CY_USER_06
APIV2 gateway version	CY_USER_10
APIV2 CornCore version	CY_USER_11
FareQuote database version	CY_USER_12

Problems found: The current general report is a customisation of a predefined TestDirector reports. In order to visualize any new field that might be added it is necessary to add them into the fields to visualize in the customisation part for the reports.

6. **Re-use some parts of VAPI-XP scripts code (generic code, templates, dll…).**

Status: Done

Performance: TestSet fields are used to share parameters and the scripts will be called through the run function of the TDHelper (object for the VAPI-XP scripts)

Alternative solutions: In the next version of TestDirector new features to allow library creation with VAPI-XP will be added. But now, the TestSet fields and the run function of the TDHelper object can be used as a work around solution.

Note: Mercury response: "Adding a library in VAPI-XP is not available in TD right now. This is a known issue and ER with ID 14025 has already been raised in our system that will be reviewed by our R&D department. This will be fixed for future release of TD as it is not implemented yet. You may use this ID to follow up on the ER with us in the future" However, for WinRunner automated tests, it is possible to create compiled modules (library) and save them in TD.

7. Automatically attach some result files in the attachment part of the 'run test' in the 'Test Lab'.

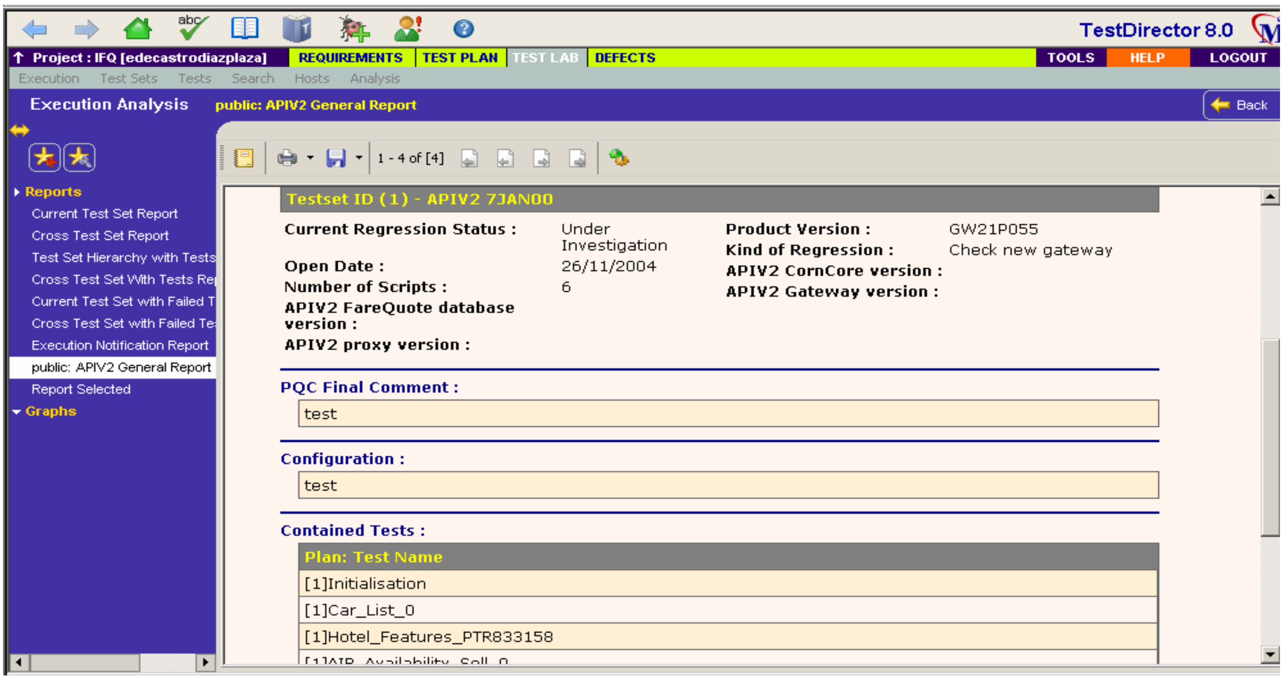
Status: Done.
Performance:

a) It is necessary to execute the code after the execution of the test cause in the workflow there is no part of code that is executed immediately after a test execution.

b) Since it would be a waste of time to write the same code at the end of all scripts and it is not possible to create libraries for VAPI-XP scripts, as Mercury Support confirmed, the solution found consist in:
   - Save the context variables needed in TestSet fields
   - Call a template code
   - Read the context variables of the caller from the TestSet fields.
   - Execute the code with the context variables of the caller test

8. Create a general report with all the mandatory data (similar to the general report in Lotus Notes).

Status: Done.
Performance: The first was to study all the different reports that TestDirector generates in each step: Requirements, TestPlan, TestLab and Defects. Afterwards, it was necessary a customisation to get with a similar format than the general report created in Lotus Notes. To customise: Configure report and subreport > Select Cros-TestSet Execution Report > Select the required fields in Custom Field Layout.


TestDirector 8.0
Project : IFQ [edecastrodiazplaza]
REQUIREMENTS
TEST PLAN
TEST LAB
DEFECTS
TOOLS
HELP
LOGOUT
Execution
Test Sets
Tests
Search
Hosts
Analysis
Execution Analysis
public: APIV2 General Report
Back
1 - 4 of [4]
Reports
Current Test Set Report
Cross Test Set Report
Test Set Hierarchy with Tests
Cross Test Set With Tests Re
Current Test Set with Failed T
Cross Test Set with Failed Te
Execution Notification Report
public: APIV2 General Report
Report Selected
Graphs

Testset ID (1) - APIV2 7JAN00

Current Regression Status :	Under Investigation	**Product Version :**	GW21P055
Open Date :	26/11/2004	**Kind of Regression :**	Check new gateway
Number of Scripts :	6	**APIV2 CornCore version :**	
APIV2 FareQuote database version :		**APIV2 Gateway version :**	
APIV2 proxy version :			

PQC Final Comment :

test

Configuration :

test

Contained Tests :

Plan: Test Name
[1]Initialisation
[1]Car_List_0
[1]Hotel_Features_PTR833158
[1]AIR_Availability_Sell_0

Alternatives solutions:

a) With OTA (Open Test Arquitecture) in TestDirector is possible to recollect the necessary data and to generate a webpage, but it is better to use TestDirector features than add extra customisation that might bring problem for next TestDirector versions.

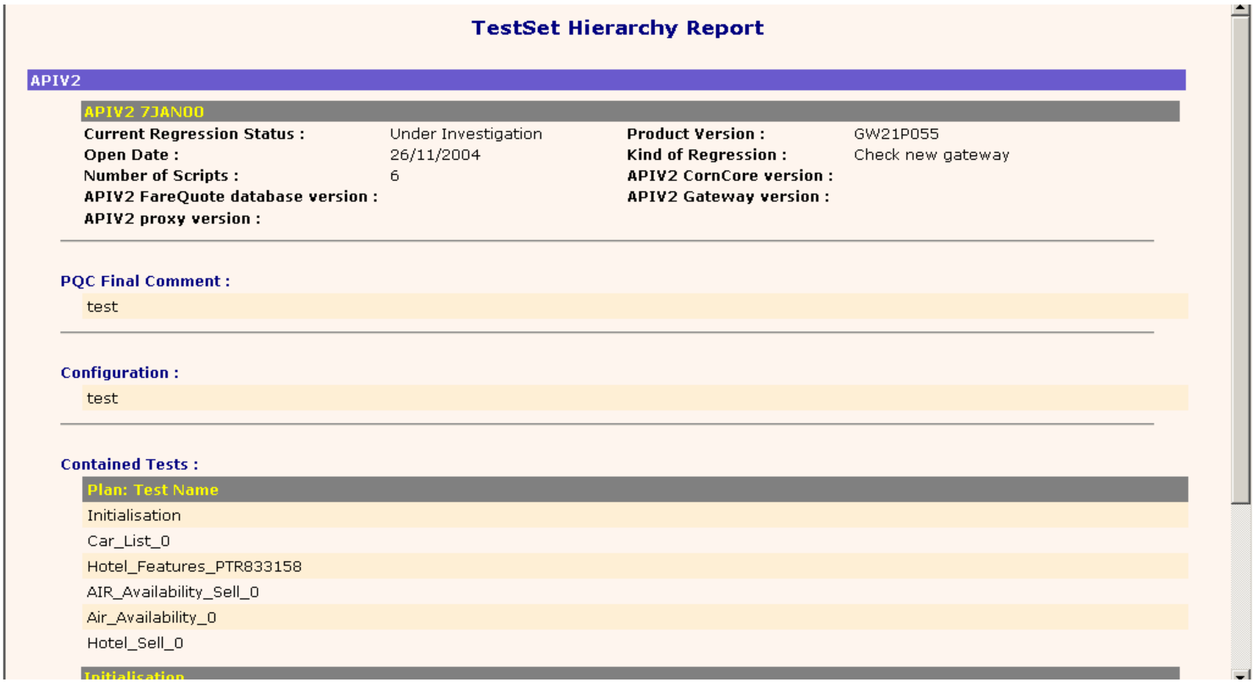
b) Another possibility would be to make changes in the report’s XML document to create a totally customised report. However, Mercury says that it is not recommended nor supported to make any modification in those templates. Anyway, the template location is the following:

Local drive --> Inetpub --> <virtual directory> TDBIN --> Reports

c) It is also possible to buy TestDirector Advanced Reports add-in which enables the customisation and design reports according to specific needs. However, the report already obtained is good enough and it doesn’t seem to be cost-effective to purchase this add-in.

9. **<u>Generate a Web page with the same data present in the general report</u>**.

Status: Done.
Performance: TestDirector allow saving a report in html.



TestSet Hierarchy Report

APIV2

APIV2 7JAN00

Current Regression Status :	Under Investigation	**Product Version :**	GW21P055
Open Date :	26/11/2004	**Kind of Regression :**	Check new gateway
Number of Scripts :	6	**APIV2 CornCore version :**	
APIV2 FareQuote database version :		**APIV2 Gateway version :**	
APIV2 proxy version :			

PQC Final Comment :

test

Configuration :

test

Contained Tests :

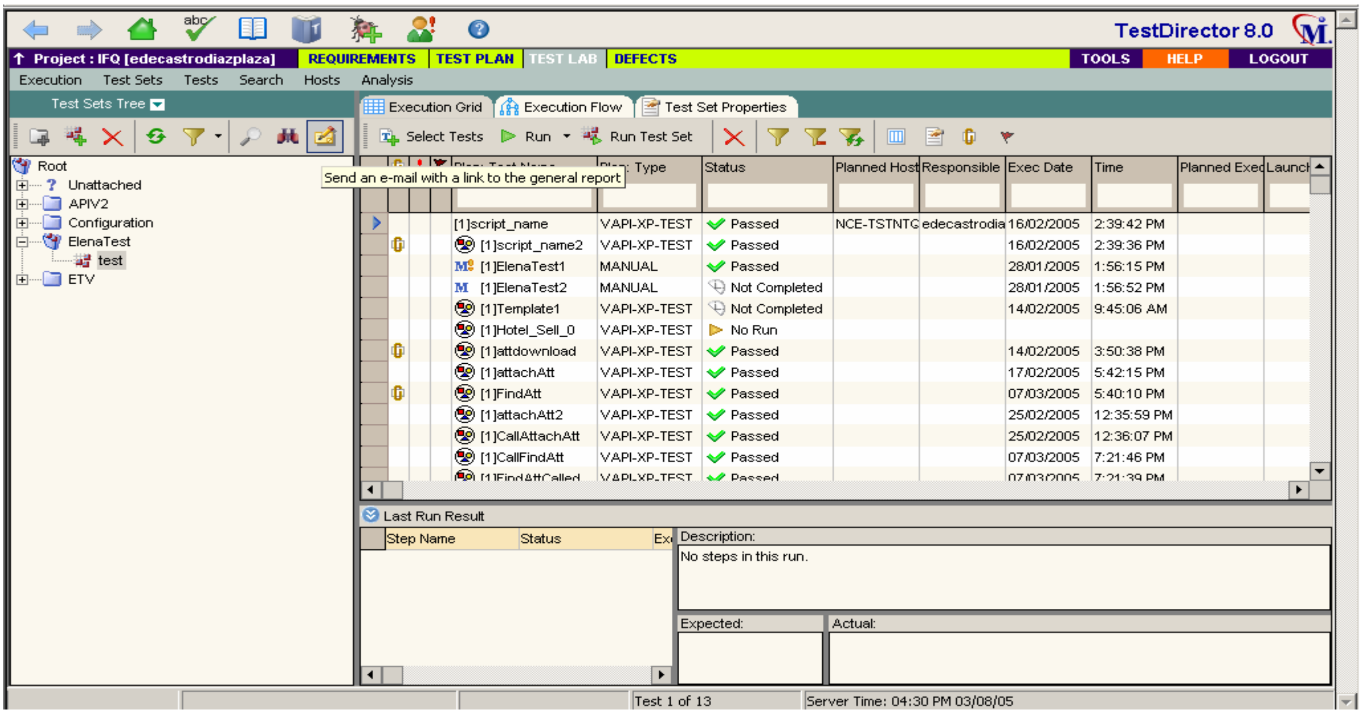
Plan: Test Name
Initialisation
Car_List_0
Hotel_Features_PTR833158
AIR_Availability_Sell_0
Air_Availability_0
Hotel_Sell_0

Initialisation

10. Send the general report by mail.

Status: Done.

Performance: A button has been created in the TestLab and it will send an e-mail to the specified persons with a link to the website where the project report is posted.



Send an e-mail with a link to the general report

Plan: Test Name	Plan: Type	Status	Planned Host	Responsible	Exec Date	Time	Planned Exec	Launch
[1]script_name	VAPI-XP-TEST	✓ Passed	NCE-TSTNTG	edecastrodia	16/02/2005	2:39:42 PM		
[1]script_name2	VAPI-XP-TEST	✓ Passed			16/02/2005	2:39:36 PM		
[1]ElenaTest1	MANUAL	✓ Passed			28/01/2005	1:56:15 PM		
[1]ElenaTest2	MANUAL	Not Completed			28/01/2005	1:56:52 PM		
[1]Template1	VAPI-XP-TEST	Not Completed			14/02/2005	9:45:06 AM		
[1]Hotel_Sell_0	VAPI-XP-TEST	▷ No Run						
[1]attdownload	VAPI-XP-TEST	✓ Passed			14/02/2005	3:50:38 PM		
[1]attachAtt	VAPI-XP-TEST	✓ Passed			17/02/2005	5:42:15 PM		
[1]FindAtt	VAPI-XP-TEST	✓ Passed			07/03/2005	5:40:10 PM		
[1]attachAtt2	VAPI-XP-TEST	✓ Passed			25/02/2005	12:35:59 PM		
[1]CallAttachAtt	VAPI-XP-TEST	✓ Passed			25/02/2005	12:36:07 PM		
[1]CallFindAtt	VAPI-XP-TEST	✓ Passed			07/03/2005	7:21:46 PM		
[1]FindAttCalled	VAPI-XP-TEST	✓ Passed			07/03/2005	7:21:39 PM		

Step Name	Status	Ex

Description:
No steps in this run.

Expected:

Actual:

Test 1 of 13

Server Time: 04:30 PM 03/08/05

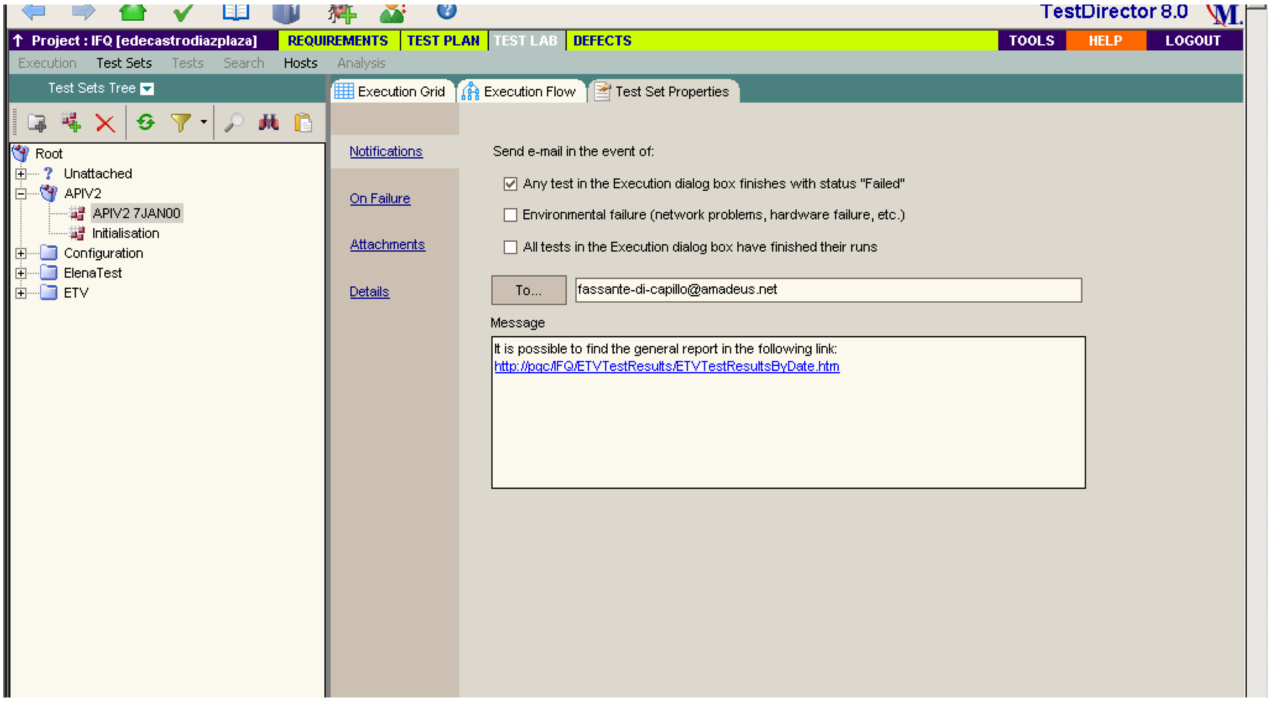
Alternative solution:

a) It is possible to program in the workflow a function that after a concrete event sends an email to specific users and with specific information. The problem here is that the information sent should be an OTA object and the general report is not. For more information about how to send a mail from the workflow look to the OTA examples file included in the appendix.

b) Use the settings in the TestSetProperties of the Test Lab

TestLab > Test Set Properties > Notifications. It is possible to send an email for the events of:

a) Any test in the execution dialog box finishes with status “failed”
b) Environment failure (network problems, hardware failure, etc…)
c) All tests in the execution dialog box have finished their runs



Send e-mail in the event of:

☑ Any test in the Execution dialog box finishes with status "Failed"

□ Environmental failure (network problems, hardware failure, etc.)

□ All tests in the Execution dialog box have finished their runs

To... fassante-di-capillo@amadeus.net

Message

It is possible to find the general report in the following link:
http://pqc/IFQ/ETVTestResults/ETVTestResultsByDate.htm

The mail contains a small report of what happened and a direct link to the test itself in TestDirector.

11. <u>Verify if it is mandatory to have SourceSafe in a local PC to extract a file (in SourceSafe).</u>

It is not necessary to have SourceSafe installed in the client machine due to it is possible to add a version control in TestDirector. It enables to keep track of the changes made to the testing information in a TestDirector project. It allows the check in and check out of the tests, displays a history of versions, and get a previous version of a test.

To integrate version control with TestDirector, a third party version control tool has to be installed, as well as the TestDirector Version Control add-in.

Version Control Add-ins:

- **<u>Microsoft Visual SourceSafe Version Control Add-in</u>**

Enables TestDirector to work with Microsoft Visual SourceSafe, allowing the version control on TestDirector tests. The Microsoft Visual SourceSafe Version Control Add-in enables version control project in TestDirector. It is possible to update and revise WinRunner, QuickTest Professional, Astra QuickTest, or VAPI-XP tests, while maintaining previous versions of each test. This allows keeping track of the changes made to each test in a TestDirector project, see how and when a test was modified, or return to a previous version of the test.

Installation Instructions:

1. Install Microsoft Visual SourceSafe server on the TestDirector server machine.

2. Uninstall any previous versions of this add-in. To uninstall, choose **Start > Settings > Control Panel > Add/Remove Programs** and follow the instructions on the screen.

3. Click **Download Add-in** to download and install this add-in to the TestDirector server machine. Note that to install this add-in, it is necessary to log in with administrator privileges.

- **Rational ClearCase Version Control Add-in**

Enables TestDirector to work with Rational ClearCase, allowing the version control on the TestDirector tests.

- **TestDirector Version Control Third Party Prep Add-in**

Prepares TestDirector before a third party version control tool is installed.

6.4) Conclusions

It has been demonstrated that the migration is possible and all the features have been implemented in the test management platform. This is going to be used for IFQ team and probably extended to many other teams in Amadeus

It has been a very stimulating project because it was a big responsibility to decide whether it is possible for a team to use a tool or not. There were no similar works done in the company so that I could take some hints and therefore everything had to be done by reading carefully the tools manuals and the Mercury support webpage to take ideas about how to solve specific customisation requirements.

It has been also interesting that one has to understand first what the customer need and then find alternatives to solve it.

7) General conclusions

I have enjoyed doing both projects because it was a new field for me and it was quite challenging that there was uncertainty about whether Microsoft Project Server 2003 was enable to handle all Amadeus employees’ use and whether the migration from Lotus Notes to TestDirector was possible or not.

It has been very interesting to get to know the latest test methodologies, the newest tools (WinRunner, LoadRunner and TestDirector), new programming languages (VBScript, TSL) and a new architecture (OTA) in relatively short period of time.

The whole project has involved not only programming or development but also I had to be in contact with the clients in order to be sure that the tests checked everything requested and that the new implementation matched with their expectations. It has been a valuable experience to realize how these things work in the real world. It is necessary to understand what the client wants and find a technically possible way to solve it. I also found it interesting to be in contact with the developers of the products I was using (Mercury support) and be able to discuss with them about customisation possibilities.

In general it has been a good professional experience and I have appreciated the two teams’ environment, within which I have also learned a lot.

8) Acknowledgments

I want to give special thanks to Philippe Bernard for helping me and guiding me always that I needed to look for information or had some troubles in the development of both projects.

I have to mention my gratitude to the IFQ team, Isabel Alexandre, Alain Ballester and Federic Assante di Capillo for explaining me how the department works and give me advises in how to approach the project.

Thanks to my professor in KTH, Christian Schulte for being in contact with me even though I was abroad and guiding me in a proper writing of my thesis.

9) Bibliography

9.1) TestDirector 8.0 Books On Line [online] Available from: http://ncesetec2/TDBIN/Help/Books/onlinedoc.htm [Accessed March 2005]

9.2) Knowledge base database and user forums in the Mercury Support website [online] Available from: http://support.mercury.com/ [Accessed March 2005]

9.3) MSDN, Visual Basic Scripting [online] Available from: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/vtoriVBScript.asp [Accessed March 2005]

9.4) VBS Tutorial [online] Available from: http://www.freenetpages.co.uk/hp/alan.gauld/tutcont.htm [Accessed March 2005]

9.5) Funcionamiento de una DLL [online] Available from: http://www.zator.com/Cpp/E1_4_4b0.htm [Accessed March 2005]

9.6) Information resource for software testers [online] Available from:

9.7) http://www.testingfaqs.org/ [Accessed March 2005]

9.8) Software Testing Tools [online] Available from: http://66.102.9.104/search?q=cache:IHnDesxcN34J:www.cs.uku.fi/research/Teho/SoftwareTestingTools.pdf+thesis+winrunner+loadrunner+testdirector&hl=es [Accessed March 2005]

9.9) Testing Term Definitions [online] Available from: http://www.olenick.com/html/glossary.html [Accessed November 2005]

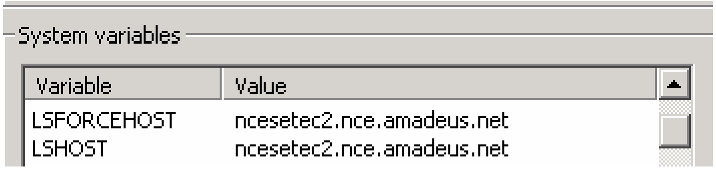
9.10) Methodologies for Automated Testing [online] Available from: http://www.sqa-test.com/method.html [Accessed November 2005]

10) Appendix

10.1) Spring project

Machine	Software configuration
Terminal server	Terminal service MS Project 2003 (“Always show full menus” checked) LoadRunner Load Generator (“Enable Terminal Services” checked) WinRunner (WebTest add-on + GUI map file per test) + patched files: *mmalloc_logic.dll* *mosifs32.dll* *thrdutil.dll* *windde32.dll* *wnrpc32.dll* *xdr.dll* wrun.ini (configuration file)
Workstation	MS Project 2003 (“Always show full menus” checked) WinRunner (WebTest add-on + GUI map file per test) + patched files: *mmalloc_logic.dll* *mosifs32.dll* *thrdutil.dll* *windde32.dll* *wnrpc32.dll* *xdr.dll* wrun.ini (configuration file) LoadRunner Load Generator
Controller	LoadRunner Controller Terminal service client (one must open **one more** terminal session client than Vusers! This is for the mdrv process)

Access to the WinRunner license server:



Variable	Value
LSFORCEHOST	ncesetec2.nce.amadeus.net
LSHOST	ncesetec2.nce.amadeus.net

SQL Database

- Load the MSP “template” plans
- Backup the database prior testing
- restrict access to “test” users only
- run the tests
- Restore the database after testing

- Plans loaded into the database for the performance tests:
  - 10 of them are about 500 tasks with 30 resources assigned to these tasks
  - 50 of them are about 200 tasks with about 8 resources assigned

10.2) DEV-SPL-TEC-TES Department

The Technical Study department mission is to provide information and make recommendations in response to requested studies and to find synergy and reuse of existing solutions between different parties within the company. Basically:

- Provides support for studies (mainly technical) that are requested by Amadeus management.
- Acts as an interface between Marketing, Development and Data Processing
- Coordinates new technical initiatives and architecture proposals
- Contributes to technical design of new products / products architectures / general technical architecture
- Follows up the implementation of the architectures to obtain the practical view

10.3) Spring project tests code

```
##########################################################################
#
#######           first script
##########################################################################
#
##############  START
#     start_transaction ("UpdatePProgress");
#     end_transaction ("UpdatePProgress", LR_AUTO);
##############    END
      wait (30);
      for(i=0;i<999;i++);

# Transactions declaration

      declare_transaction ("OpenPlan");
      declare_transaction ("SaveAs");
      declare_transaction ("PublishAllInformation");
      declare_transaction ("Login");
      declare_transaction ("TaskDisplay");
      declare_transaction ("UpdateAll");
      declare_transaction ("OpenPlan2");
      declare_transaction ("UpdatePProgress");
      declare_transaction ("UpdateMP");


# Vuser

      lr_whoami (group, scenario, vuser);
      output_message ("Virtual User: " & vuser);
      Num = vuser;
      if (Num == -1)
      {
            srand (get_time ());
      Num = int (rand()*6);
            Num = 1;
      }
      else
      {
            Num = Num-1;
      }

      cont2 = 99;#this counter should go decreasing so that the last
project plan created
                        # is displayed the first in the web access tool.
      var = 100 - cont2;
    num_iterations_big_loop = 2;

########
######## Big loop
########
 while (var <= num_iterations_big_loop)
 {

# Press "Start"
      set_window ("Shell_TrayWnd", 500);
      button_press ("Start");

# Launch MProject for SPRING-Num through a command in "Run"
      set_window ("BaseBar", 500);
      toolbar_select_item ("ToolbarWindow32_1", "Run...");
      set_window ("Run", 500);
      edit_set ("Open:_1", sprintf("Winproj.exe /s
\"http://nceprojspring.nce.amadeus.net/projectserver\" /u \"SPRING-%i\" /p
\"\" ",Num));
      button_press ("OK");


# Select bar menu in MProject
      win_wait_info ("Microsoft Project - Project1","enabled",1,500);
      set_window ("Microsoft Project - Project1", 500);
      obj_mouse_click ("Menu Bar", 47, 9, LEFT);

# Select File
      win_wait_info ("File_1","displayed",1,500);
      win_mouse_click ("File_1", 13, 31);

# Open from Microsoft Office Project Server
      win_wait_info ("Open from Microsoft Office Project
Server","displayed",1,500);
      set_window ("Open from Microsoft Office Project Server", 44);
      obj_click_on_text ("JWinproj-GridClass",sprintf("AAASPRING%i-TEST-
0",Num),LEFT);
      win_mouse_click ("Open from Microsoft Office Project Server", 236,
383);
      win_mouse_click ("Open from Microsoft Office Project Server", 400,
396);

##############  START
      start_transaction ("OpenPlan");

################################
### Save as with different name
################################

# Microsoft Project - SPRING-TEST.Published [Read-Only]
      i = 0;
      while (win_wait_info ("Microsoft Project - SPRINGX-TEST-n.Published
[Read-Only]","displayed",1,1000)==E_NOT_FOUND && i<6)
      {           i++;
      }
      end_transaction ("OpenPlan", LR_PASS);
##############    END
```

```
      set_window ("Microsoft Project - SPRINGX-TEST-n.Published [Read-
Only]", 500);
      obj_mouse_click ("Menu Bar", 46, 8, LEFT);

# File_1
      i = 0;
      while (win_wait_info ("File_1","displayed",1,1000)==E_NOT_FOUND &&
i<6)
      {           i++;
      }
      win_mouse_click ("File_1", 63, 126);

# Save to Project Server
      win_wait_info ("Save to Project Server","displayed",1,500);
      set_window ("Save to Project Server", 500);
      obj_exists ("JWinproj-Edit",500);
      # The plan created will be called: SPRINGX-TEST-X
      obj_type ("JWinproj-Edit","<kDel_E>");
#     wait(2);
      obj_type ("JWinproj-Edit",sprintf("<kDel_E>AAASPRING%i\-TEST\-
%i<kDel_E>",Num,cont2));
      win_mouse_click ("Save to Project Server", 339, 273);

##############  START
      start_transaction ("SaveAs");



# Microsoft Project - SPRING-TEST-.Published
      i = 0;
      while (win_wait_info("Microsoft Project - SPRINGX-TEST-
n.Published","enabled",1,500)==E_NOT_FOUND && i<20)
      {           i++;
      }

      end_transaction ("SaveAs", LR_PASS);
##############    END

      set_window ("Microsoft Project - SPRINGX-TEST-n.Published", 500);
      obj_wait_info("Menu Bar","enabled",1,500);
      obj_mouse_click ("Menu Bar", 358, 12, LEFT);

      win_wait_info ("Collaborate_1","enabled",1,600);
      win_click_on_text ("Collaborate_1","Publish",FALSE,LEFT);
      win_wait_info ("Publish_0","enabled",1,600);
      win_click_on_text ("Publish_0","Republish",FALSE,LEFT);

# Microsoft Office Project

      if (win_exists ("Microsoft Office Project",10) == 0)
      {
            set_window ("Microsoft Office Project", 500);
            button_press ("OK");
      }

# Republish Assignments
      win_mouse_click ("Republish Assignments", 406, 364);

      wait (4);

##############  START
      start_transaction ("PublishAllInformation");
```

```
###############################
### Close MProject
##############################

# Microsoft Project - SPRING-TEST-23-10.Published
      i = 0;
      while ((win_wait_info("Microsoft Project - SPRINGX-TEST-
n.Published","enabled",1,500) == E_NOT_FOUND) && i<20)
      {           i++;
      }

      end_transaction ("PublishAllInformation", LR_PASS);
##############    END

# Microsoft Project
      win_wait_info ("Microsoft Project","enabled",1,500);
      set_window ("Microsoft Project");
      win_close ("Microsoft Project");

      win_wait_info ("Microsoft Office Project","enabled",1,500);
      set_window ("Microsoft Office Project", 500);
      button_press ("No");

      win_wait_info ("Microsoft Project","displayed",0,500);
##########################################################################
#
#######           end first script
##########################################################################
#

##########################################################################
#
#######           second script
##########################################################################
#

# Variables
      cont = 0;
      num_iterations = 10;

#     Now the script checks if there are tasks before so it is not
neccessary to wait here
#     wait (10) ;

 while (cont < num_iterations)
 {
# Shell_TrayWnd
      set_window ("Shell_TrayWnd", 500);
      button_press ("Start");

# BaseBar
      set_window ("BaseBar", 500);
      toolbar_select_item ("ToolbarWindow32_1", "Programs;Internet
Explorer");

# Browser Main Window

      set_window ("Browser Main Window", 500);
      wait (2);
#     edit_set ("browser_main_edit_location_0", "<kDel_E>");
      obj_type ("browser_main_edit_location_0", "<kDel_E>");
      obj_type ("browser_main_edit_location_0",
"<kDel_E>http://nceprojspring.nce.amadeus.net/projectserver/lgnps.asp");
#     edit_set ("browser_main_edit_location_0",
"<kDel_E>http://nceprojspring.nce.amadeus.net/projectserver/lgnps.asp");
      edit_set_selection ("browser_main_edit_location_0", 0, 0, 0, 45);
      obj_type ("browser_main_edit_location_0","<kReturn>");


##############
### Log in
##############
# Microsoft Office Project Web Access 2003 Logon - nceprojspring.nce.ama
      set_window("Microsoft Office Project Web Access 2003 Logon - 
nceprojspring.nce.ama",500);
      edit_set("userName",sprintf("spring-%i-%i",Num,cont));
      button_press("Go");
##############  START
      start_transaction ("Login");


# Microsoft Office Project Web Access 2003 - nceprojspring.nce.amadeus
      win_wait_info ("Microsoft Office Project Web Access 2003 - 
nceprojspring.nce.amadeus","enabled",1,500);

      end_transaction ("Login", LR_PASS);
##############    END
      set_window ("Microsoft Office Project Web Access 2003 - 
nceprojspring.nce.amadeus", 500);
      while (win_click_on_text ("Microsoft Office Project Web Access 2003 
- nceprojspring.nce.amadeus","no",FALSE,LEFT)==0)
      {
            # There are no new tasks assigned
            web_link_click("Home");
            wait (3);
      }

# Microsoft Office Project Web Access 2003 - nceprojspring.nce.amadeus
      win_wait_info ("Microsoft Office Project Web Access 2003 - 
nceprojspring.nce.amadeus","enabled",1,500);
      set_window("Microsoft Office Project Web Access 2003 - 
nceprojspring.nce.amadeus",500);
      web_link_click("Tasks");

##############    START
      start_transaction ("TaskDisplay");

      set_window ("Microsoft Office Project Web Access 2003 - 
nceprojspring.nce.amadeus", 500);

      end_transaction ("TaskDisplay", LR_PASS);
##############    END

##################################
### Insert notes and time report
##################################
      set_window("Microsoft Office Project Web Access 2003 - 
nceprojspring.nce.amadeus",500);
      # Waiting for the grid to be displayed
    obj_exists ("Insert Notes",500);
#     wait (2);

      obj_mouse_click ("JWinproj-GridClass_0", 169, 95, LEFT);
      web_sync(8);
      button_press("Insert Notes");

# Project Web Access Assignment Notes
      set_window("Project Web Access Assignment Notes",500);
      edit_set("idNoteCurrent",sprintf("[SPRING-%i-%i]try\r\n",Num,cont));
      button_press("OK");

# html_frame
      set_window ("html_frame", 500);
      obj_mouse_click ("JWinproj-GridClass_1", 111, 93, LEFT);
      type ("22");
      obj_mouse_click ("JWinproj-GridClass_1", 159, 94, LEFT);

# Microsoft Office Project Web Access 2003 - nceprojspring.nce.amadeus
      set_window("Microsoft Office Project Web Access 2003 -
nceprojspring.nce.amadeus",500);
      button_press("Insert Notes");

# Project Web Access Assignment Notes
      set_window("Project Web Access Assignment Notes",500);
      edit_set("idNoteCurrent",sprintf("[SPRING-%i-%i]try2\r\n[SPRING-%i-
%i]try",Num,cont,Num,cont));
      button_press("OK");

# Microsoft Office Project Web Access 2003 - nceprojspring.nce.amadeus
      set_window("Microsoft Office Project Web Access 2003 -
nceprojspring.nce.amadeus",500);
      button_press("Update All");
############## START
      start_transaction ("UpdateAll");

# Project Web Access
      win_wait_info("Project Web Access","enabled",1,500);

      end_transaction ("UpdateAll", LR_PASS);
##############    END
      set_window("Project Web Access",500);
      button_press("OK");

##############
### Log off
##############

# Microsoft Office Project Web Access 2003 - nceprojspring.nce.amadeus
      win_wait_info("Microsoft Office Project Web Access 2003 -
nceprojspring.nce.amadeus","enabled",1,500);
      set_window("Microsoft Office Project Web Access 2003 -
nceprojspring.nce.amadeus",500);
      web_link_click("Log Off");

# Browser Main Window
      win_close ("Browser Main Window");

      cont = cont + 1;
 }#end loop
##########################################################################
#
#######           end second script
##########################################################################
#

##########################################################################
#
#######           third script
##########################################################################
#

#     The file has been saved in web test mode so it doesn´t need to load
```

```
GUI files
#    but to run it, it should have winrunner in web test mode and GUI file
per test mode.

#######################################
### Open MProject with a concrete user
#######################################

 # Microsoft Project
      set_window ("Shell_TrayWnd", 500);
      button_press ("Start");

# BaseBar
      set_window ("BaseBar", 500);
      toolbar_select_item ("ToolbarWindow32_1", "Run...");
      set_window ("Run", 500);
      edit_set ("Open:_1", sprintf("Winproj.exe /s
\"http://nceprojspring.nce.amadeus.net/projectserver\" /u \"SPRING-%i\" /p
\"\" ",Num));
      button_press ("OK");


# Microsoft Project
      win_wait_info ("Microsoft Project - Project1","enabled",1,500);
      set_window("Microsoft Project - Project1");
      obj_mouse_click ("Menu Bar", 40, 11, LEFT);

# File_1
      win_mouse_click ("File_1", 20, 37);

# Open from Microsoft Office Project Server
      win_wait_info ("Open from Microsoft Office Project
Server","enabled",1,500);
      set_window ("Open from Microsoft Office Project Server", 500);
      obj_click_on_text ("JWinproj-GridClass",sprintf("AAASPRING%i-TEST-
%i",Num,cont2),LEFT);
      win_mouse_click ("Open from Microsoft Office Project Server", 406,
395);
##############  START
      start_transaction ("OpenPlan2");


# Microsoft Project - SPRING-TEST2.Published
      win_wait_info ("Microsoft Project - SPRINGX-TEST-
n.Published","enabled",1,800);

      end_transaction ("OpenPlan2", LR_PASS);
##############    END

      set_window ("Microsoft Project - SPRINGX-TEST-n.Published", 500);
      obj_mouse_click ("Menu Bar", 357, 4, LEFT);

# Collaborate_1
      win_wait_info ("Collaborate_1","enabled",1,900);
      win_mouse_click ("Collaborate_1", 46, 59);

##############  START
      start_transaction ("UpdatePProgress");

      set_window ("Shell DocObject View_3", 500);
      while (win_click_on_text ("Shell DocObject
View_3","all",TRUE,LEFT)!=0)
      {
            # There are no new tasks assigned
            wait (2);
      }

      end_transaction ("UpdatePProgress", LR_PASS);
##############    END

      wait (1);
      set_window ("Shell DocObject View_3", 500);
      button_press ("Accept_all");

      win_wait_info ("Microsoft Project - SPRINGX-TEST-
n.Published","enabled",1,500);
      set_window ("Microsoft Project - SPRINGX-TEST-n.Published", 500);
      button_press ("Update");

      set_window ("Shell DocObject View_3", 500);
      button_press ("UpdateC");

##############    START
      start_transaction ("UpdateMP");


# Microsoft Office Project
      win_wait_info ("Microsoft Office Project","enabled",1,900);

      end_transaction ("UpdateMP", LR_PASS);
##############    END

      set_window ("Microsoft Office Project", 500);
      button_press ("OK");

# Project Web Access -- Web Page Dialog
      win_wait_info ("Project Web Access -- Web Page
Dialog","enabled",1,900);
      set_window ("Project Web Access -- Web Page Dialog", 500);
      obj_mouse_click ("Internet Explorer_Server", 252, 101, LEFT);

# Microsoft Project - SPRING-TEST2.Published
      win_wait_info ("Microsoft Project - SPRINGX-TEST-
n.Published","enabled",1,999);
      win_close ("Microsoft Project - SPRINGX-TEST-n.Published");
      wait (1);

##########################################################################
#
#######           End third script
##########################################################################
#
      cont2 = cont2 - 1;
      var = 100 - cont2;
} # End big loop
########
######## End Big loop
########
```

10.4) IFQ Team

Within the Product Quality Control department, the Internet and Front Office Quality team is in charge of:

- Implementation and operation of Quality Regression Test platforms for Amadeus Central System components and Client Products (e.g. Vista, Cruise, Tempo, APIs...)
- Develop and maintain internal tools and processes to help the automation of tasks for all the teams of the Product Quality Control department
- Study, suggest and help implement QRT infrastructure for new Amadeus products.

10.5) TestDirector customisation code

These are the files that have been implemented into the customisation section:
The .vbs files are templates and example of templates to be used in the Test Lab module. These files are defined in TestDirector as VAPI-XP scripts.

1) The following files: attachAtt2.vbs, FindAtt2.vbs and script_name2 are templates.

2) The following files: CallAttachAtt.vbs, CallFindAtt.vbs and script_name2 are only examples of how the templates should be called and what parameters have to be sent.

3) attachAtt.vbs, attdownload.vbs and FindAtt.vbs contain the code itself before having been converted into templates or buttons.

4) Test Lab Module script is the code corresponding to the workflow in that part. In this part the buttons are implemented.

The code of attachAtt2 and CallAttachAtt is enclosed as a example of how templates work. Below, the code of the Test Lab Module in the workflow is listed to give a better understanding of the context variables and how does it looks like.

attachAtt2.vbs

```
' attachAtt2 [VBScript]
' Created by TestDirector
' 17/02/2005 17:32:42
' ====================================================


' ----------------------------------------------------
' Main Test Function
' Debug - Boolean. Equals to false if running in [Test Mode] : reporting
to TestDirector
' CurrentTestSet - [OTA COM Library].TestSet.
' CurrentTest - [OTA COM Library].TSTest.
' CurrentRun - [OTA COM Library].Run.
' ----------------------------------------------------
Sub Test_Main(Debug, CurrentTestSet, CurrentTest, CurrentRun)
  ' *** VBScript Limitation ! ***
  ' "On Error Resume Next" statement suppresses run-time script errors.
  ' To handle run-time error in a right way, you need to put "If
Err.Number <> 0 Then"
  ' after each line of code that can cause such a run-time error.
  On Error Resume Next

  ' clear output window
  TDOutput.Clear

   'Similar to attachAtt but this one is implemented to be a template and
   ' being executed through CallAttachAtt
   Dim testname, testid, objTSTest, objLastRun
   'Get the file name stored in CY_USER_01
   testName = CurrentTestSet.Field("CY_USER_01")
   msgbox "Get value of Parameter1: " & CurrentTestSet.Field("CY_USER_01")

   'Get the file name stored in CY_USER_01
   testId = CurrentTestSet.Field("CY_USER_13")
   msgbox "Get value of Parameter2: " & CurrentTestSet.Field("CY_USER_13")


    Set objTSTest = CurrentTestSet.TSTestFactory.Item(testId)
    Set objLastRun = objTSTest.LastRun
    msgbox "testName " & testName

    Dim attAddresstref, attNametref, faddresstref, fpathtref
    Dim attAddresstres, attNametres, faddresstres, fpathtres

    If objLastRun.Field("RN_ATTACHMENT") <> "Y" Then

      msgbox "adding tref attachment"

      dim attfact 'attachment factory
      dim att 'attachment

      set attfact = objLastRun.Attachments

      set att = attfact.AddItem(NULL)

      fpathtref = "\\nce-tstntq-d3\APIv2Tests\V_GW21P055\TRANSREF"
      faddresstref = fpathtref & "\" & testName & ".tref"

      att.FileName = faddresstref
      att.Type = 1
      att.Post

      'getting path for the tref attachment
      attNametref = att.DirectLink
      attAddresstref    =    "\\ncesetec2\TD_Dir\PQCIFQ\IFQ0\attach\"    &
attNametref


      msgbox "adding tres attachment"

      set att = attfact.AddItem(NULL)

      fpathtres = "\\nce-tstntq-d3\APIv2Tests\V_GW21P055\TRANSRESULT"
      faddresstres = fpathtres & "\" & testName & ".tres"

      att.FileName = faddresstres
      att.Type = 1
      att.Post

      'getting path for tres attachment
      attNametres = att.DirectLink
      attAddresstres    =    "\\ncesetec2\TD_Dir\PQCIFQ\IFQ0\attach\"    &
attNametres
```

```
      'Launching windiff with the att
      'msgbox "Launching windiff with tres and tref"
      'Set WshShell = CreateObject("Wscript.Shell")
      'intReturn = WshShell.Run("windiff " & attAddresstref & " " &
attAddresstres, 1, True)

    End If
    set attfact=nothing
    set att = nothing

  If Not Debug Then
  End If
  ' handle run-time errors
  If Err.Number <> 0 Then
    TDOutput.Print "Run-time error [" & Err.Number & "] : " &
Err.Description
    ' update execution status in "Test" mode
    If Not Debug Then
      CurrentRun.Status = "Failed"
      CurrentTest.Status = "Failed"
    End If
  End If
End Sub
```

CallAttachAtt

```
' CallAttachAtt [VBScript]
' Created by TestDirector
' 17/02/2005 17:30:03
' ====================================================


' ----------------------------------------------------
' Main Test Function
' Debug - Boolean. Equals to false if running in [Test Mode] : reporting
to TestDirector
' CurrentTestSet - [OTA COM Library].TestSet.
' CurrentTest - [OTA COM Library].TSTest.
' CurrentRun - [OTA COM Library].Run.
' ----------------------------------------------------
Sub Test_Main(Debug, CurrentTestSet, CurrentTest, CurrentRun)
  ' *** VBScript Limitation ! ***
  ' "On Error Resume Next" statement suppresses run-time script errors.
  ' To handle run-time error in a right way, you need to put "If
Err.Number <> 0 Then"
  ' after each line of code that can cause such a run-time error.
  On Error Resume Next

  ' clear output window
  TDOutput.Clear


  'The name of the script is stored in CY_USER_01 to be able to be checked
  '  by all the tests within this TestSet
  CurrentTestSet.Field("CY_USER_01")= CurrentTest.testname
  CurrentTestSet.Post()
  msgbox "new value of Parameter1: " & CurrentTestSet.Field("CY_USER_01")


  'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  'ANOTHER PARAMETER AND CALL TO attachAtt

  'The id of the script is stored in CY_USER_13 to be able to be checked
  '  by all the tests within this TestSet
  CurrentTestSet.Field("CY_USER_13")= CurrentTest.id
  CurrentTestSet.Post()
  msgbox "new value of Parameter2: " & CurrentTestSet.Field("CY_USER_13")
  ' TODO: put your code here

  ' ejecutar script_name2 desde este script que es script_name
  msgbox "call attachAtt"
  res = TDHelper.RunTest ("attachAtt2", 1, "test")

  If Not Debug Then
  End If
  ' handle run-time errors
  If Err.Number <> 0 Then
    TDOutput.Print  "Run-time  error  ["  &  Err.Number  &  "]  :  "  &
Err.Description
    ' update execution status in "Test" mode
    If Not Debug Then
      CurrentRun.Status = "Failed"
      CurrentTest.Status = "Failed"
    End If
  End If
End Sub
```

TestLab Module in the workflow

```
Sub SendLinkToGeneralReport()
On Error Resume Next

   TDConnection.SendMail "edecastrodiazplaza@amadeus.net",
"edecastrodiazplaza@amadeus.net", "Information sent from TestDirector", "Find the general
report: http://pqc/IFQ/ETVTestResults/ETVTestResultsByDate.htm ", NULL, "HTML"
   'SendMail "from","to","Subject","Message",array_of_att,"HTML"/"Text"
   MsgBox "Mail Sent"

    On Error GoTo 0
End Sub

'This function sends the test object selected and any attachments that
' had been added to it in the TestPlan part.
'The 3rd argument of the Mail function can have the following values:
'  TDMAIL_ATTACHMENT [1]
'  TDMAIL_HISTORY [2]
'  TDMAIL_TEXT [4]
'  TDMAIL_DES_STEP [8]
'  TDMAIL_COVER_TEST [16]
'  TDMAIL_SINGLEMAIL [32]
'  TDMAIL_COMMENT_AS_BODY [64]
Sub SendTest (iObjectId, strTo, strCc, strSubject, strComment)
'This function is not in use now but it is an example about
' how to send an object by email.
On Error Resume Next
 Dim objTestFactory, objTest
     Set objTestFactory = TDConnection.TestFactory
     Set objTest = objTestFactory.Item (iObjectId)
     objTest.Mail strTo, strCc, 1, strSubject, strComment
     Set objTest = nothing
     Set objTestFactory = nothing
 PrintError "SendTest"
 On Error GoTo 0
End Sub

Sub AddAttsAfterEachRun()
On Error Resume Next
Dim iTestSetId, iTSTestId, objTestSet, objTSTest, objLastRun
Dim testName
Dim attAddresstref, attNametref, faddresstref, fpathtref
Dim attAddresstres, attNametres, faddresstres, fpathtres

    iTestSetId = TestSetTest_Fields("TC_CYCLE_ID").Value
    iTSTestID = TestSetTest_Fields("TC_TEST_ID").Value
    Set objTestSet = TDConnection.TestSetFactory.Item(iTestSetId)
    Set objTSTest = objTestSet.TSTestFactory.Item(iTSTestID)
    Set objLastRun = objTSTest.LastRun
    testName = objTSTest.TestName
    msgbox "testName " & testName

    If objLastRun.Field("RN_ATTACHMENT") <> "Y" Then

      msgbox "adding tref attachment"

      dim attfact 'attachment factory
      dim att 'attachment

      set attfact = objLastRun.Attachments

      set att = attfact.AddItem(NULL)

      fpathtref = "\\nce-tstntq-d3\APIv2Tests\V_GW21P055\TRANSREF"
      faddresstref = fpathtref & "\" & testName & ".tref"

      att.FileName = faddresstref
      att.Type = 1
      att.Post

      'getting path for the tref attachment
      attNametref = att.DirectLink
      attAddresstref = "\\ncesetec2\TD_Dir\PQCIFQ\IFQ0\attach\" & attNametref


      msgbox "adding tres attachment"

      set att = attfact.AddItem(NULL)

      fpathtres = "\\nce-tstntq-d3\APIv2Tests\V_GW21P055\TRANSRESULT"
      faddresstres = fpathtres & "\" & testName & ".tres"
      att.FileName = faddresstres
      att.Type = 1
      att.Post

      'getting path for tres attachment
      attNametres = att.DirectLink
      attAddresstres = "\\ncesetec2\TD_Dir\PQCIFQ\IFQ0\attach\" & attNametres

      'Launching windiff with the att
      msgbox "Launching windiff with tres and tref"
      Set WshShell = CreateObject("Wscript.Shell")
      intReturn = WshShell.Run("windiff " & attAddresstref & " " & attAddresstres, 1, True)

    End If
    set attfact=nothing
    set att = nothing

    Set objTestSet = Nothing
    Set objTSTest = Nothing
    Set objLastRun = Nothing
PrintError "AddAttsAfterEachRun"
On Error GoTo 0
End Sub

Sub LaunchWindiffWithAtts()
On Error Resume Next
Dim iTestSetId, iTSTestId, objTestSet, objTSTest, objLastRun
Dim testName
Dim attAddresstref, attNametref, faddresstref, fpathtref
Dim attAddresstres, attNametres, faddresstres, fpathtres

    iTestSetId = TestSetTest_Fields("TC_CYCLE_ID").Value
    iTSTestID = TestSetTest_Fields("TC_TEST_ID").Value
    Set objTestSet = TDConnection.TestSetFactory.Item(iTestSetId)
    Set objTSTest = objTestSet.TSTestFactory.Item(iTSTestID)
    Set objLastRun = objTSTest.LastRun
    testName = objTSTest.TestName
    'msgbox "testName " & testName

    msgbox "Launching windiff with tres and tref"
    If objLastRun.Field("RN_ATTACHMENT") = "Y" Then

      'msgbox "adding tref attachment"

      Dim attfact 'attachment factory
      Dim att 'attachment
      set attfact = objLastRun.Attachments

      Dim alist
      Set alist = attfact.NewList("")
      MsgBox "The test " & testName & " has " & alist.Count & " attachments"
      'msgbox "id " & objLastRun.id

      Dim vartres, vartref, foundtres, foundtref
      foundtres = false
      foundtref = false
      vartres = "RUN_" & objLastRun.id & "_" & testName & ".tres"
      vartref = "RUN_" & objLastRun.id & "_" & testName & ".tref"

      'attAddresstref = alist.Item(alist.Count).FileName

      For each att IN alist
        if att.Name = vartres then
          'load function has a bug and downloads the file to a temp folder
          ' but that is ok for this function. That path is retrieved to
          ' attAddresstref
          att.Load True, "C:\"
          attAddresstref = att.FileName
          msgbox att.Name & " found"
          foundtref = True
        end if
        if att.Name = vartref then
          att.Load True, "C:\"
          attAddresstres = att.FileName
          msgbox att.Name & " found"
          foundtres = True
        end if
      Next

      if foundtref and foundtres then
         'Launching windiff with the att tres y tref
          Set WshShell = CreateObject("Wscript.Shell")
          intReturn = WshShell.Run("windiff " & attAddresstref & " " & attAddresstres, 1,
True)
   else
          msgbox "tres and/or tref are not attached to the last run of the file"
   end if
    else
      msgbox "There are no attachments"

    End If
    set attfact=nothing
    set att = nothing

    Set objTestSet = Nothing
    Set objTSTest = Nothing
    Set objLastRun = Nothing
    Set WshShell = Nothing
PrintError "LaunchWindiffWithAtts"
On Error GoTo 0
End Sub
```

```
Function TestLab_ActionCanExecute(ActionName)

  'On Error Resume Next
  dim WshShell
  dim iObjectId, strTo, strCc, strSubject, strComment
  TestLab_ActionCanExecute = Project_DefaultRes
  TestLab_ActionCanExecute= true

  if ActionName = "SendMail" then
    '1st way to do it
    SendLinkToGeneralReport

    '2nd way
    'iObjectId = TestSetTest_Fields.Field("TC_TEST_ID").Value
    'strTo = "edecastrodiazplaza@amadeus.net"
    'strCc = ""
    'strSubject = "Information sent test"
    'strComment = "Find the general report:
http://pqc/IFQ/ETVTestResults/ETVTestResultsByDate.htm "
    'SendTest iObjectId, strTo, strCc, strSubject, strComment
    'msgbox "the 3rd e-mail to: " & strTo & " has been sent"
  end if

  If ActionName = "launch_windiff_with_atts" Then
    LaunchWindiffWithAtts
  End IF

  On Error GoTo 0
End Function




Sub TestLab_Attachment_New(Attachment)
  On Error Resume Next
      'msgbox "Sub TestLab_Attachment_New(Attachment)"
  On Error GoTo 0
End Sub

Sub TestLab_RunTestSet(Tests)
  On Error Resume Next

  On Error GoTo 0
End Sub

Sub TestLab_TestSet_MoveTo
  On Error Resume Next

  On Error GoTo 0
End Sub

Function TestLab_InitNewTask(Items, NewTask)
```

```
  On Error Resume Next

  TestLab_InitNewTask = Project_DefaultRes
  On Error GoTo 0
End Function

Sub TestLab_TestSetTests_FieldChange(FieldName)
  On Error Resume Next

  On Error GoTo 0
End Sub

Sub TestLab_TestSet_FieldChange(FieldName)
  On Error Resume Next

  On Error GoTo 0
End Sub

Sub TestLab_TestSetTests_MoveTo
  On Error Resume Next
    'msgbox "Sub TestLab_TestSetTests_MoveTo"
  On Error GoTo 0
End Sub

Function TestLab_TestSetTests_FieldCanChange(FieldName, NewValue)
  On Error Resume Next

  TestLab_TestSetTests_FieldCanChange = Project_DefaultRes
  On Error GoTo 0
End Function

Sub TestLab_RunTests(Tests)
  On Error Resume Next
     'msgbox "Sub TestLab_RunTests(Tests)"
  On Error GoTo 0
End Sub

Sub TestLab_TestSet_AfterPost
  On Error Resume Next
     'msgbox "Sub TestLab_TestSet_AfterPost"
  On Error GoTo 0
End Sub

Function TestLab_TestSet_CanPost
  On Error Resume Next
   'msgbox "Function TestLab_TestSet_CanPost"
  TestLab_TestSet_CanPost = Project_DefaultRes
  On Error GoTo 0
End Function

Function TestLab_Attachment_CanPost(Attachment)
  On Error Resume Next
```

```
    'msgbox "Function TestLab_Attachment_CanPost(Attachment)"
  TestLab_Attachment_CanPost = Project_DefaultRes
  On Error GoTo 0
End Function

Sub TestLab_DialogBox(DialogBoxName, IsOpen)
  On Error Resume Next

  On Error GoTo 0
End Sub

Sub TestLab_ExitModule
  On Error Resume Next

  On Error GoTo 0
End Sub

Sub TestLab_EnterModule
  On Error Resume Next

  On Error GoTo 0
End Sub
```

10.6) OTA Examples:

10.6.1) I. Rules of Thumb

Use Error Handling in all procedures and functions.

See: Error Handling

Optimize your code and minimize its size.

Use all the features VBScript provides.

See: Code Optimization

Use TestDirector API whenever possible instead of external objects, files, registry.

See: Using TestDirector API (OTA) in scripts

Explicitly initialize the values for all field properties for all fields.

See: Workflow Objects

10.6.2) 1. Error Handling

Use “On Error Resume Next” statement at the beginning of each procedure and function.

Use “On Error GoTo 0” at the end of each procedure or function

Show errors to the user in some standard message box

Compare the following code fragments:

If Bug_Fields(“BG_SEVERITY”).Value = “4-Urgent” Then Bug_Fields(“BG_PRIORITY”).Value = “4-Urgent” End If	On Error Resume Next ... If Bug_Fields(“BG_SEVERITY”).Value = “4-Urgent” Then Bug_Fields(“BG_PRIORITY”).Value = “4-Urgent” End If ... PrintError “SomeSub” On Error GoTo 0

The both code fragments will work the same way when no problems arise during the code execution. But what happens if, for example, the user has no permissions to modify BG_PRIORITY field? The left fragment will not update the field and will cause the browser crash. The second will not update the field, but it will show the correct error to the user and the browser will not crash.

See: Code Templates -> Error Handling

10.6.3) 2. Code Optimization

Use procedures and functions instead of the redundant code
Use Switch statement instead of repetitive ElseIf statements
Combine all values that require the same code to be executed to the same Case statement

- **Use procedures and functions instead of the redundant code**

The following common tasks are an example of a good subject for separate function/procedure:

- Setting field properties;
- Setting up fields on the form;
- Setting list dependencies;
- Any procedure that works with TD API (like: send mail)

- **Use Switch statement instead of repetitive ElseIf statements**

The rule of thumb here can be: if there are 2 or more ElseIf conditions, use Switch statement.

- **Combine all values that require the same code to be executed to the same Case statement**

VBScript allows you to put several values into one Case statement. Example:
In the following example the same code [Code A] should be executed when the {Variable} value is X or Y. Instead of creating 2 cases (code on the left side), you can put X and Y into 1 case (code on the right side).

Select Case {Variable}
Case X
    [Code A]
Case Y
    [Code A]
Case Z
    [Code B]
End Select
	Select Case {Variable}
	Case X, Y
	    [Code A]
	Case Z
	    [Code B]
	End Select

10.6.4) 3. Using TestDirector API (OTA) in Scripts

Use a predefined TDConnection object to get the current session.
Use standard OTA Interfaces. Avoid use of the Command interface.
Use TestDirector favorites to store user and group-related data or data common for all users.
Use mailing methods available in OTA to send the custom mails to the users.

- **Use a predefined TDConnection object to get the current session.**

When you use OTA from some external tool (like VB), the first step for any application that uses OTA is to create the instance of the TDConnection object, initialize the connection to the server, and connect to the database.
But in the Workflow there is the predefined TDConnection object (in this case TDConnection is not only class name, but also the name of the global variable that contains the instance of TDConnection), which points the same session in which the current user works.
This means that access to all TestDirector collections and objects is always available from any place in the Workflow.

See: Code Templates -> Working With TestDirector API -> Getting the current connection (current session)

- **Use standard OTA Interfaces. Avoid use of the Command interface.**

The Command object allows execution of any query directly against the database (using the QueryExecute method). This seems easy solution, but it has two major disadvantages:

1. It avoids the whole TestDirector logic (permissions, verification of data integrity, etc.), and
2. It works directly against database, avoiding TestDirector optimizations, which reduces the performance.

This refers to the usage of QueryExecute for SELECT statements. But an even more dangerous situation is when the Workflow updates or deletes records from tables using the QueryExecute method. This may lead to database corruption, unexpected errors, etc.

From the other side, it is always possible to avoid the use of the Command interface and to use standard TestDirector interfaces instead (enough to say that even TestDirector R&D does not use the QueryExecute method).

See: Code Templates -> Working With TestDirector API

- **Use TestDirector favorites to store user and group-related data or data common for all users.**

TestDirector favorites are the universal place for storage of any user or group-related information, as well as the information common for all users.

The following are the examples when TestDirector favorites can be used:

- Store the last used values for the fields per user.
- Store the up-to-date defaults (like “current version” or “current build”) for fields for all users.

The advantages of the settings usage over the external files/registry are

- The TestDirector settings are not dependant on the machine from which the user connects to TestDirector.
- They use all TestDirector optimizations.
- They are “native” to TestDirector, and do not need any additional objects, DLLs, etc.

The OTA has two classes that allow access to the favorites:

- The CommonSettings class allows access/modification of the TestDirector Public Favorites.
- The UserSettings class allows access/modification of the TestDirector Private Favorites.

Both classes can be retrieved from the TDConnection object.

See: Code Templates -> Working With TestDirector API -> Keeping Last Used Value In Fields

- **Use mailing methods available in OTA to send the custom mails to the users.**

OTA allows access to TestDirector mailing, which allows you to:

- Create custom conditions that cannot be implemented using the automatic notification system of TestDirector.
- Change the Subject or the text of the e-mail.
- Send an e-mail to the specific TestDirector groups or TestDirector users.
- Send the e-mail from the specific user, rather then “admin” as automatic mail notification does.

The mailing methods are available from any TestDirector object (like Defect, Test, etc.) or directly from the TDConnection object. Using the Mail method from the TestDirector object you can send the e-mail that contains that object and your custom subject/text. Using the Mail method from the TDConnection object allows you to send any custom mail.

See: Code Templates -> Working With TestDirector API -> Sending an E-Mail from the Workflow

10.6.5) 4. Workflow Objects

Use {Object}_Fields("{Field_Name}") to access field by name.
Use loop on {Object}_Fields.FieldById(i) to access all fields in the collection.
Set the IsVisible property before setting the IsRequired or IsReadOnly property of the field.
Reset the layout for all fields before setting the fields layout (PageNo and ViewOrder).

- **Use {Object}_Fields("{Field_Name}") to access field by name.**

Use loop on {Object}_Fields.FieldById(i) to access all fields in the collection.
Both methods allow you to work with the fields of the "current object."
The current object can be defined in the following way:

- **Current object type:** The object type for which the current event works. Almost each event points to the object type for which the fields can be retreived. For example, in Defects_Bug_... events, only Bug_Fields can be retreived; in TestPlan_DesignStep_... events, only DesignStep_Fields can be retrieved.
- **Focused item:** From all objects of the collection defined by event, only the fields of the currently focused object can be retrieved. For example, the test on which the cursor is placed or the current run in manual runner.

To retrieve the fields of other objects of the same/other object type, use TD API (OTA).
The first statement *{Object}_Fields("{Field_Name}")* can be used to retrieve any particular field by name.
The second statement *{Object}_Fields.FieldById(i)* is needed to go over all fields of the current object. For example, to reset the fields order.

See: Code Templates -> Working With the Fields

- **Set the IsVisible property before setting the IsRequired or IsReadOnly property of the field.**

Setting the IsRequired or IsReadOnly property for the field not visible in the UI is meaningless and is ignored by TestDirector. So it is important to ensure that the field is visible before setting any of these properties.

See: Code Templates -> Working With the Fields -> Setting Field Properties

- **Reset the layout for all fields before setting the fields' layout (PageNo and ViewOrder).**

Since the fields have some default predefined order, it is important to reset this order before defining the new, custom one.
Consider the following example:

```
Bug_Fields("BG_SEVERITY").ViewOrder = 1
Bug_Fields("BG_PRIORITY").ViewOrder = 2
```

```
For i=0 To Bug_Fields.Count
    Bug_Fields.FieldById(i).ViewOrder = 100
Next
Bug_Fields("BG_SEVERITY").ViewOrder = 1
Bug_Fields("BG_ PRIORITY").ViewOrder = 2
```

In the first example, the ViewOrder is set only for BG_SEVERITY and BG_PRIORITY fields. The ViewOrder of other fields is unknown (what if ViewOrder of some other field is 1 as well? In this case, BG_ PRIORITY will be third and not the second field on the form), so you do not actually know how the fields will appear on the form. In the second example, the ViewOrder of all the fields is reset to some big value, which ensures that BG_SEVERITY and BG_PRIORITY fields will be indeed the first on the form.

See: Code Templates -> Working With the Fields -> Resetting Properties of All Fields

10.6.6) II. Code Templates

1. Error Handling

1.1 Show the Standard Error to the User

Purpose: The following procedure shows the standard error to the user.
Code Location: The code should be added once to each Workflow script (Defects, Test Plan, etc.).
Templates Used: None
Arguments:
strFunctionName - The name of the function or procedure in which the error have happened
Return Value: None

Code Template:

```
Sub PrintError(strFunctionName)
        If Err.Number <> 0 Then
   MsgBox "Error #" & Err.Number & ": " & Err.Description, _
                vbOKOnly+vbCritical, _
                "Workflow Error in Function " & strFunctionName
    End If
End Sub
```

Template 1.1

Usage Example: See the template in 2.1.

1.2 Error Handling in Procedures and Functions

Purpose: The following code ensures the correct error handling in procedures and functions.
Code Location: The code should be added to each function and procedure.
Templates Used: Template 1.1
Arguments: Not relevant
Return Value: Not relevant

Code Template:

```
Function|Sub {Function|Sub_Name}()
        On Error Resume Next
        [Your code here]
        PrintError "{Function|Sub_Name}"
        On Error GoTo 0
End Function|Sub
```

Template 1.2

Usage Example: See the templates below.

2. Working with the Fields

2.1 Setting Field Properties

Solution 1: Setting all field properties
Purpose: The code below allows the setting of all field properties.
Code Location: The code should be added to the code of each Workflow script (Defects, Test Plan, etc.) for each object (for example, in Test Plan script the separate procedures for Test and Design Steps should be added). The {Object} should be replaced with the name of the object (i.e., Bug, Test, etc.)

Templates Used: Template 1.1 and 1.2
Arguments:
strFieldName - The name of the field for which the properties should be set
blnIsVisible - The value of the IsVisible property of the field to be set
blnIsReadOnly - The value of the IsReadOnly property of the field to be set
blnIsRequired - The value of the IsRequired property of the field to be set
intPageNo – The value of the PageNo property of the field to be set
intViewOrder – The value of the ViewOrder property of the field to be set
Return Value: None

Code Template:

```
Sub {Object}_SetFieldProp(strFieldName,
blnIsVisible, blnIsReadOnly, blnIsRequired,
        intPageNo, intViewOrder)
On Error Resume Next
        With {Object}_Fields(strFieldName)
        .IsVisible = blnIsVisible
        .IsReadOnly = blnIsReadOnly
        .IsRequired = blnIsRequired
        .PageNo = intPageNo
        .ViewOrder = intViewOrder
        End With
        PrintError "{Object}_SetFieldProp"
On Error GoTo 0
End Sub
```

Template 2.1.1

Usage Example: For example, if you have created the procedure Sub Bug_SetFieldProp using this template, you can use it anywhere in the script like:

```
        Bug_SetFieldProp "BG_USER_01", True, False, False, 2, 1
```

<u>**Solution 2: Setting field flags and view properties separately**</u>
Purpose: Sometimes it is more convenient to separate the field flags (IsVisible, IsRequired, IsReadOnly) and field view properties (PageNo and ViewOrder). The code below allows for setting the field flags and view properties separately.
Code Location: The code should be added to the code of each Workflow script (Defects, Test Plan, etc.) for each object (for example in Test Plan script the separate procedures for Test and Design Steps should be added). The {Object} should be replaced with the name of the object (i.e., Bug, Test, etc.)
Templates Used: Template 1.1 and 1.2
Arguments:
strFieldName - The name of the field for which the properties should be set
blnIsVisible - The value of the IsVisible property of the field to be set
blnIsReadOnly - The value of the IsReadOnly property of the field to be set
blnIsRequired - The value of the IsRequired property of the field to be set
intPageNo – The value of the PageNo property of the field to be set
intViewOrder – The value of the ViewOrder property of the field to be set
Return Value: None

Code Template:

```
Sub {Object}_SetFieldFlags(strFieldName,
blnIsVisible, blnIsReadOnly, blnIsRequired)
On Error Resume Next
        With {Object}_Fields(strFieldName)
        .IsVisible = blnIsVisible
        .IsReadOnly = blnIsReadOnly
        .IsRequired = blnIsRequired
        End With
        PrintError "{Object}_ SetFieldFlags"
On Error GoTo 0
        End Sub
```

Template 2.1.2

```
Sub {Object}_SetFieldView(            strFieldName,
            intPageNo, intViewOrder)
On Error Resume Next
            With {Object}_Fields(strFieldName)
            .PageNo = intPageNo
            .ViewOrder = intViewOrder
            End With
            PrintError "{Object}_ SetFieldView"
On Error GoTo 0
End Sub
```

Template 2.1.3

Usage Example: See the examples for templates 2.1.1 and 2.2.

2.2 Resetting Properties of All Fields

Solution 1: Resetting field properties in loop

Purpose: Before setting the field flags or layout properties, it's recommended to reset the properties of all fields in the collection to ensure that all the properties are set to correct values (See: Workflow Objects in Rules of Thumb section). The code below shows how to reset properties of all fields in the collection.

Code Location: The code should be added in each relevant procedure or function (when the reset of field properties is needed). Alternatively, it can be put to the separate procedure. In this case the separate procedures are needed for different object types (for example in Test Plan script the separate procedures for Test and Design Steps should be added).

The {Object} should be replaced with the name of the object (ie: Bug, Test, etc.), {Property} should be replaced with the name of field property (ie: IsRequired, PageNo, etc) and the {Value} should be replaced with property value.

Templates Used: None

Arguments: Not relevant

Return Value: Not relevant

Code Template:

```
For i=0 To {Object}_Fields.Count
            {Object}_Fields.FieldById(i).{Property} = {Value}
Next
```

Template 2.2.1

Usage Example: in the following example, the IsVisible property of all defect fields is reset:

```
            For i=0 To Bug_Fields.Count
                        Bug_Fields.FieldById(i).IsVisible = False
            Next
```

Solution 2: Calling function that resets field properties in loop

Purpose: Same as above. In this case the template procedures will be used to reset the field properties (the {Procedure} should be replaced with the name of the procedure used to reset the fields. The name of the field can be retrieved using

Code Location: Same as above.

Templates Used: templates 2.1.1, 2.1.2, 2.1.3

Arguments: Not relevant

Return Value: Not relevant

Code Template:

```
For i=0 To {Object}_Fields.Count
            {Procedure} {Object}_Fields.FieldById(i).FieldName[, ...]
Next
```

Template 2.2.2

Usage Example: in the following example, the procedure Bug_SetFieldView was created from template 2.1.3. The procedure is used to reset the layout of all fields:

```
        For i=0 To Bug_Fields.Count
                Bug_SetFieldView Bug_Fields.FieldById(i).FieldName, 100, 100
        Next
```

2.3 Setting Field Layout on Form

Purpose: Organize fields order on forms and define pages. Using the template, you can specify the names of the field just in the same way as you want the fields to appear on page. To define several pages, call the procedure with different arrays.
Code Location: The code should be added to the code of each workflow script (Defects, Test Plan, etc.), for each object (for example in Test Plan script the separate procedures for Test and Design Steps should be added). The {Object} should be replace with the name of the object (ie: Bug, Test, etc.). The {Form} should be replace with the name of the form (ie: Add Defect, Defect Details, etc.)
Templates Used: Template 1.1, 1.2, 2.1.3
Arguments:
strFieldArray - the name of the field for which the properties should be set.
intPageNo – the value of the PageNo property of the field to be set.
Return Value: None
Note: It’s recommended to reset the layout of all the fields before setting the new layout (ie: to set PageNo and ViewOrder properties to some big values, like 100, and to set IsVisible, IsRequired and IsReadOnly properties of all fields to False. See: Workflow Objects in Rules of Thumb section and 2.2 code template)

Code Template:

```
Sub {Form}_SetPageFieldsOrder(strFieldArray, intPageNo)
On Error Resume Next
        For i=0 To Ubound(strFieldArray)
        {Object}_SetFieldView strFieldArray(i), intPageNo, i + 1
        Next
PrintError "{Form}_SetPageFieldsOrder"
On Error GoTo 0
End Sub
```

Template 2.3

Usage Example: for example you want to organize the fields on the Add Defect form, so that all relevant system fields would be on first page, and all other fields would be on the second page, the AddDefect_SetPageFieldsOrder procedure created by template can be used in the following way:

```
        Dim strFieldArray1, strFieldArray2
        ‘ First page
        strFieldArray1 = Array(“BG_DETECTED_BY”, “BG_DETECTION_DATE”, _
          “BG_STATUS”, “BG_SEVERITY”, _
          “BG_PROJECT”, “BG_DETECTION_VERSION”, _
          “BG_SUBJECT”, “BG_REPRODUCIBLE”)
        AddDefect_SetPageFieldsOrder strFieldArray1, 1
        ‘ Second page
        strFieldArray2 = Array(“BG_USER_01”, “BG_USER_02”, _
          “BG_USER_03”, “BG_USER_04”)
        AddDefect_SetPageFieldsOrder strFieldArray2, 2
```

2.4 Setting List Dependencies

Purpose: The code below allows to set up the value for some field (ie: Dependant Field) according to the value of the other field (ie: Master Field).
Code Location: It should be added to the code of each workflow script (Defects, Test Plan, etc.). Also the different procedures should be created for all pairs of field that have different logic. The {Object} should be replace with the name of the object (ie: Bug, Test, etc.)

Templates Used: templates 2.1.1, 2.1.2
Arguments:

strMasterField - the name of the field which defines the list for the dependant field
strDepenadantField – the name of the field for which the list is set according to master field.
Return Value: None

Code Template:

```
Sub {Object}_SetList(strMasterField, strDependantField)
On Error Resume Next
        Select Case {Object}_Fields(strMasterField).Value
        Case "{Value1}"
          {Object}_Fields(strDependantField).List = Lists("{List1}")
        …
        Case Else
          {Object}_Fields(strDependantField).List = Lists("{ListN}")
        End Select
PrintError "{Object}_SetList"
On Error GoTo 0
End Sub
```

Template 2.4

Usage Example: For example if you've created the procedure Sub Bug_SetList using this template, and you want to set the list for BG_CLOSING_VERSION, BG_DETECTION_VERSION, BG_PLANNED_CLOSING_VER fields according to the value of the BG_PROJECT field, you can use the following code anywhere in the script:

```
        Bug_SetList "BG_PROJECT", "BG_CLOSING_VERSION"
        Bug_SetList "BG_PROJECT", "BG_ DETECTION _VERSION"
        Bug_SetList "BG_PROJECT", "BG_PLANNED_CLOSING_VER"
```

2.5 Ensuring That User Updates Some Field When Another Field Is Changed

Purpose: The code below allows to ensure that some field ("dependant field") is updated when another field ("master field") is changed to some value. For example: when the Status (aka "master field") field is changed to "Fixed", we want to ensure that the user updates R&D Comments (aka "dependant field").
Code Location: It should be added to the code of each workflow script (Defects, Test Plan, etc.), separately for each object. The {Mode} should be replaced with the name of the mode (TestPLan, TestLab, etc.); the {Object} should be replaced with object name (Bug, Test, etc.). {MasterField} should be replaced with the name of the "master field" (or with field label for flag names); {DependantField} should be replaced with the name of the "dependant field" (or with field label for flag names). See the example below for correct template usage.
Templates Used: None
Arguments: None
Return Value: None
Note: Since the template is only a part of the code, and doesn't represent a separate function or procedure, the error handling is not shown in template.
Steps:

1. Create 2 global boolean flags that will be False by default; one of them will become True when the "master field" is changed to desired value; the second will become true when the "dependant field" will be changed.
2. Update flags to False in MoveTo event
3. In FieldChange event, update the first flag to True when the "master field" is changed to desired value; update the second flag to True when the dependant field is changed
4. In CanPost event, if the "master field" was changed (ie: the first flag is True), and the second field was not changed (ie: the second flag is False), do not allow item posting.

Code Template:

```
        Dim b{MasterField}Changed, b{DependantField}Changed

        Sub {Mode}_{Object}_MoveTo()
        …
        b{MasterField}Changed = False
        b{DependantField}Changed = False
        …
        End Sub
        Sub {Mode}_{Object}_FieldChange(FieldName)
        …
        Select Case FieldName
        …
        Case "{MasterField}"
          If {Object}_Fields("{MasterField}").Value = {Value} Then
        b{MasterField}Changed = True
          End If
        Case "{DependantField}"
        b{DependantField}Changed = True
        …
        End Select
        …
        End Sub

        Function {Mode}_{Object}_CanPost()
        …
        If b{MasterField}Changed And _
  Not b{DependantField}Changed Then
        MsgBox "Please update <" & _
            {Object}_Fields("{DependantField}").FieldLabel & _
            "> field.", vbCritical + vbOKOnly, _
            "Workflow - Field Change Verification"
        {Mode}_{Object}_CanPost = False
Else
        {Mode}_{Object}_CanPost = True
End If
        …
        End Function
```

Template 2.5

Usage Example: In the following example, when the Status field is changed to "Fixed", the scripts ensures that R&D Comments field was updated as well.

```
Dim bStatusChanged, bRDCommentsChanged

Sub Defects_Bug_MoveTo()
    bStatusChanged = False
    bRDCommentsChanged = False
End Sub

Sub Defects_Bug_FieldChange(FieldName)
    Select Case FieldName
    Case "BG_STATUS"
        If Bug_Fields("BG_STATUS").Value = "Fixed" Then
            bStatusChanged = True
        End If
    Case "BG_DEV_COMMENTS"
        bRDCommentsChanged = True
    End Select
End Sub

Function Defects_Bug_CanPost()
    If bStatusChanged And Not bRDCommentsChanged Then
        MsgBox "Please update <" & _
             Bug_Fields("BG_DEV_COMMENTS").FieldLabel & _
             "> field.", vbCritical + vbOKOnly, _
             "Workflow – Field Change Verification"
        Defects_Bug_CanPost = False
    Else
        Defects_Bug_CanPost = True
    End If
End Function
```

2.6 Check That the Object Is Not Yet Submitted to the Project ("new object")

Purpose: In many cases you need to verify if the object was not yet submitted to the project. The following code allows to perform such verification. For example: you want to verify that the new defect is always submitted with the status 'New'; on test creation you want to fill some user-defined field; etc.
Code Location: The code should be added in each relevant procedure or function (when such verification is needed). {Object} should be changed with the name of the object for which the verification is needed (for example: Bug, Test, etc.); the {Object ID Field} should be changed to the name of the field stores the ID of the object (for example: BG_BUG_ID, TS_TEST_ID, etc.).
Templates Used: None
Arguments: Not relevant
Return Value: Not relevant

Code Template: Check that the object ID field is empty

```
If {Object}_Fields("{Object ID Field}").Value = "" Then
```

Template 2.6

Usage Example: The example below checks if the defect is "new object", and doesn't allow to post such defect id its status is not "New":

```
        Sub Defects_Bug_CanPost()
        On Error Resume Next
        …
                If Bug_Fields("BG_BUG_ID").Value = "" And _
                  Bug_Fields("BG_STATUS").Value <> "New" Then
                        MsgBox "The <Status> of the new bug should be 'New'."
                        Defects_Bug_CanPost = False
                End If
        …
PrintError "Defects_Bug_CanPost"
On Error GoTo 0
        End Sub
```

2.7 Revert the Field to the Old Value On Update

Purpose: The code below allows to revert the value of the field to the old value, when some condition is not satisfied. For example: the Status of the bug cannot be set to Closed if Closed In Version field was not filled.
Code Location: There are 3 parts of relevant code. The first part defines the global variable that will store the original value of the field. For each field the separate variable should be defined. The second part of code should be added MoveTo event (this part preserves the original value of the field, before it was changed). The third part of code should be added to the CanPost event (it checks the condition and reverts the field value to the original value if the condition is not satisfied). All 3 parts of code should be added to each module, separately for each field and condition when the old value retrieval mechanism is required.
Templates Used: None
Arguments: Not relevant
Return Value: Not relevant

Code Template:

```
        Dim s{FieldLabel}OrigValue
        …
        Sub {Mode}_{Object}_MoveTo()
          …
  s{FieldLabel}OrigValue = {Object}_Fields("{FieldName}"). Value
          …
        End Sub
        …
        Function {Mode}_{Object}_CanPost()
          …
          If {Condition} Then
            {Object}_Fields("{FieldName}"). Value = s{FieldLabel}OrigValue
          End If
        …
        End Sub
```

Template 2.7

Usage Example: In the example below when the user sets the status of the defect to Closed, and doesn't fill the Closed In Version field, the value of the Status field is changed back to the original value:

```
        Dim sStatusOrigValue
        …
        Sub Defects_Bug_MoveTo()
          …
  sStatusOrigValue = Bug_Fields("BG_STATUS").Value
  …
End Sub
…
Function Defects_Bug_CanPost()
  …
  If Bug_Fields("BG_CLOSING_VERSION").Value = "" Then
        MsgBox "The Closed In Version value was not specified.", vbCritical
        Bug_Fields("BG_STATUS").Value = sStatusOrigValue
  End If
  …
End Function
```

2.8 Implementing "By Owner Only" by Multiple Fields

Purpose: While TestDirector allows one user to be the owner of the object, sometimes it's reasonable to have several owners of the object. For example if you want that only the user sho submitted the defect ("Detected By", BG_DETECTED_BY field), or the user who is responsible for the defect ("Assigned To", BG_RESPONSIBLE field) will be able to change its Priority and Severity. Other users can change other fields. If you will set "By Owner Only" flag for Priority and Severity fields in group permissions, only "Assigned To" user will be able to change Priority and Severity (since by default in TestDirector the owner of the defects is ("Assigned To" user). If "By Owner Only" flag for Priority and Severity fields will not be set in group permissions, all users will be able to change these fields.
The code below allows to work with multiple owners of the object.
Code Location: The function should be added to the code of each workflow script (Defects, Test Plan, etc.), for each object (for example in Test Plan script the separate procedures for Test and Design Steps should be added). The {Object} should be replace with the name of the object (ie: Bug, Test, etc.). The function should be called from {Mode}_{Object}_FieldChange or {Mode}_{Object}_CanChange events, as shown in the example.
Templates Used: Template 1.1 and 1.2
Arguments:
strFieldNames - the names of the fields that define the owner of the object. The system and user-defined fields that have the User list attached can be specified. The format of the string should be: "FIELD_NAME1;FIELD_NAME2;…"
Return Value: Boolean: True if the current user is the owner of the object; False if the current user is not the owner of the object.

Code Template:

```
Function Check{Object}Owner(strFieldNames)
On Error Resume Next
Dim strFieldArr
CheckObjectOwner = False
strFieldArr = Split(strFieldNames, ";")
For i = 0 To Ubound(strFieldArr)
If {Object}_Fields(strFieldArr(i)).Value = User.UserName Then
CheckObjectOwner = True
End If
Next
PrintError "Check{Object}Owner"
```

```
On Error GoTo 0
End Sub
```

Template 2.8

Usage Example: In the example below 3 fields define the owner of the defect: BG_DETECTED_BY, BG_RESPONSIBLE and BG_USER_01 (user-defined field with the User list attached). When the user tries to change defect's Priority (BG_PRIORITY) or Severity (BG_SEVERITY), workflow checks if the user is the owner of the defect (ie: the name of current user appears in one of 3 specified fields). If the user is not the owner of the object, the field value is reverted to the original value. Note that template 2.7 is used in this example.

```
Dim sPriorityOrigValue
Dim sSeverityOrigValue

Sub Defects_Bug_MoveTo
…
        sPriorityOrigValue = Bug_Fields("BG_PRIORITY").Value
sSeverityOrigValue = Bug_Fields("BG_SEVERITY").Value
…
End Sub

Sub Bug_FieldChange(FieldName)
Dim strOwnerFields
strOwnerFields = "BG_DETECTED_BY;BG_RESPONSIBLE;BG_USER_01"
…
        Select Case FieldName
        Case "BG_SEVERITY", "BG_PRIORITY"
          If Not CheckBugOwner(strOwnerFields) Then
                MsgBox "Only owners can change defect Severity and Priority"
                Bug_Fields("BG_PRIORITY").Value  = sPriorityOrigValue
Bug_Fields("BG_SEVERITY").Value = sSeverityOrigValue
          End If
        End Select
…
End Sub
```

3. Working with TestDirector API (OTA)

3.1 Getting the current connection (current session)

Purpose: The code below shows how to receive the current session context in workflow (the same session in which the user will work during script execution).
Code Location: The code should be added in each relevant procedure or function (when the access to OTA is required).
Templates Used: None
Arguments: Not relevant
Return Value: Not relevant

Code Template: Just refer to the **TDConnection** object

Usage Example: In the example below server time (TDConnection class property) is shown in message box:

```
MsgBox "Current time on server is: " & TDConnection.ServerTime
```

3.2 Getting current session properties

Purpose: The code below allows getting the properties of the current session (server URL, server time, domain name, project name, project type, user name, and password). Any other TDConnection property can be obtained

using the alike code. For the list of TDConnection properties, please see the help for TDConnection object in OTA Guide.
Code Location: code should be added to the appropriate function or procedure where any of these properties is needed (the properties do not depend on each other, meaning that any of the properties can be received separately).
Templates Used: None
Arguments: Not relevant
Return Value: Not relevant

Code Template:

```
        Dim varRes
        ‘ Returns the current server URL
        varRes = TDConnection.ServerName
        ‘ Returns the current server time (Date type)
        varRes = TDConnection.ServerTime
        ‘ Returns the current domain name
        varRes = TDConnection.DomainName
        ‘ Returns the current project name
        varRes = TDConnection.ProjectName
        ‘ Returns the current project type (Access, Oracle, MS SQL, Sybase)
        varRes = TDConnection.ProjectType
‘ Returns the current user name (no need to use TDConnection here – ‘ the workflow has the predefined object called User)
        varRes = User.UserName
        ‘ Returns the current user password
        varRes = TDConnection.Password
```

Template 3.2

Usage Example: In the example below, the server name is used to verify whether the user connected to the server using HTTP or HTTPS:

```
If Left(UCase(TDConnection.ServerName), 5) = “HTTPS” Then
MsgBox “You are currently connected to the server using SSL”
Else
        MsgBox “You are not using SSL”
End If
```

<u>3.3 Finding to which groups the current user belongs</u>

<u>Solution 1: Checking if the user belongs to some particular group</u>

Purpose: The code below allows checking if the current user belongs to a given group.
Code Location: code should be added to the appropriate function or procedure where it’s needed to check if the user belongs to any particular group.
Templates Used: None
Arguments: Not relevant
Return Value: Not relevant

Code Template: Just refer to the **User.IsInGroup(“Group_Name”)** property

Usage Example: The example below checks if the user belongs to TDAdmin group

```
If User.IsInGroup(“TDAdmin”) Then
        MsgBox “You are the member of TDAdmin group”
Else
        MsgBox “You are not the member of TDAdmin group”
End If
```

<u>Solution 2: Receiving the list of all the groups to which the current user belongs</u>

Purpose: The code below allows getting the list of all groups to which the current user belongs.
Code Location: code should be added to the code of each workflow script (Defects, Test Plan, etc.).
Templates Used: templates 2.1.1, 2.1.2

Arguments: None
Return Value: string that contains the names of the groups to which the user belongs. The names of the groups are separated by semicolon.

Code Template:

```
Function GetUserGroups()
Dim objCustomization, objUsers, objUser, objGroup
Dim strGroupList
On Error Resume Next
        Set objCustomization = TDConnection.Customization
        Set objUsers = objCustomization.Users
        Set objUser = objUsers.User(User.UserName)
        strGroupList = “”
        For Each objGroup In objUser.GroupsList
        strGroupList = strGroupList & “;” & objGroup.Name
        Next
        GetUserGroups = Left(strGroupList, Len(strGroupList)-1)
        Set objCustomization = Nothing
        Set objUsers = Nothing
        Set objUser = Nothing

PrintError "GetUserGroups"
On Error GoTo 0
End Function
```

Template 3.3

Note: You can use Split function in order to convert the string returned by this function into array:

```
Dim strGroupArray
strGroupArray = Split(GetUserGroups, “;”)
```

Usage Example: in the following example the list of the groups is received using the template above; the user role is later defined according to the groups. For example if the user is the member of “QA Tester” team, the user role is defined as “QA”; if the user is the member of “Managers” and “Developers” groups, then user role is defined as “R&D Manager” etc.

```
Dim strGroups, strRole
strGroups = UCase(GetUserGroups)
If InStr(strGroups, “DEVELOPER”) > 0 Then strRole = “R&D “
If InStr(strGroups, “QA”) > 0 Then strRole = strRole  & “QA “
If InStr(strGroups, “MANAGER”) > 0 Then strRole = strRole & “Manager “
strRole = Left(strRole, Len(strRole) – 1)
```

3.4 Keeping last used value in fields

Purpose: The following code template allows preserving the value of the field, and to fill the field automatically next time the field needs to be filled (even after the closure of the browser, or from different client machine). The template uses TestDirector private favorites to store the settings (which allow using the saved fields’ settings even if the user connects from different machine). The template is especially useful for Add Defect form and the Manual Runner, but can be used for any other object as well.
Code Location: code should be added to the code of each workflow script (Defects, Test Plan, etc.). Separate procedures should be created for each object for which this functionality is needed. The {Object} statement should be replaced with the name of the object for which the template is used (for example: Bug, Test, etc.).
Templates Used: templates 2.1.1, 2.1.2
Arguments:
strAction - Accepts the values “GET” and “SET”. When “GET” is specified for strAction argument, the values of the fields previously saved are retrieved (the value for the field is set only if the field is empty). When “SET” is specified for strAction argument, the values of the specified fields are stored (the second argument specifies which fields need to be saved).
strFieldArray – Relevant only when “SET” is specified as the strAction argument: the array that contains the names of the fields for which the values should be saved. The empty string can be specified for “GET” action.
Note: The second argument of the function (strFieldArray) can be defined as follows before function call:

```
        Dim strFieldArray
        strFieldArray = Array(“{Field Name1}”, “{Field Name2}”, …)
```

See also the usage example below.

Return Value: None

Code Template:

```
Sub KeepLastValue(strAction, strFieldArray) ‘ strAction = {“GET”|”SET”}
On Error Resume Next
Dim strValues, objUserSettings, strFields, strField
        If strAction = "SET" Then
        strValues = ""
        For i=0 To UBound(strFieldArray)
        If strValues <> "" Then strValues = strValues & ";"
        strValues = strValues & strFieldArray(i) & "=" & _
                 {Object}_Fields(strFieldArray(i)).Value
        Next
        End If
        Set objUserSettings = TDConnection.UserSettings
        If strAction = "SET" Then
        objUserSettings.Open ("KeepLastValue")
        objUserSettings.Value("{Object}_KeepLastValue") = strValues
        objUserSettings.Close
        End If
        If strAction = "GET" Then
        objUserSettings.Open ("KeepLastValue")
        strValues = objUserSettings.Value("{Object}_KeepLastValue")
        If strValues <> "" Then
        strFields = Split(strValues, ";")
        For i = 0 To UBound(strFields)
        strField = Split(strFields(i), "=")
        If UBound(strField) = 1 And {Object}_Fields(strField(0)).Value = "" Then
        {Object}_Fields(strField(0)).Value = strField(1)
        End If
        Next
        End If
        End If
        objUserSettings.Close
        Set objUserSettings = Nothing
PrintError “KeepLastValue(“ & strAction & “)”
On Error GoTo 0
End Sub
```

Template 3.4

Usage Example: in the following example the values of Detected in Version field (BG_DETECTION_VERSION) and Project field (BG_PROJECT) are saved when the user submits the new bug. The saved values are retrieved the next time the user opens Add Defect form.

```
        Sub Defects_Bug_New
On Error Resume Next
…
        ‘ Retrieve the values of the fields on opening of new bug
        KeepLastValue "GET", ""
        …
        PrintError “Defects_Bug_ New”
On Error GoTo 0
End Sub

Function Defects_Bug_CanPost
On Error Resume Next
Dim strFieldArray
…
        ‘ Save the values of the fields on submit of new bug
        strFieldArray = Array(“BG_DETECTION_VERSION”, "BG_PROJECT")
```

```
            If Bug_Fields("BG_BUG_ID").Value = "" Then
            KeepLastValue "SET", strFieldArray
            End If
…
PrintError "Defects_Bug_CanPost"
On Error GoTo 0
            End Function
```

3.5 Sending an E-Mail from the Workflow

Purpose: The following code template allows sending any object in TestDirector (Defect, Test, etc.) by e-mail. The code can be used to extend the automatic mail notifications with custom conditions, not available for Send All Qualified. The template uses TestDirector mailing functions.

Code Location: The code for the each object should be placed in the script of the corresponding module. For example: SendDefect function should be placed in Defects module.

Templates Used: templates 2.1.1, 2.1.2

Arguments:

iObjectId – The ID of the object that should be sent (see the note below on how to obtain the object ID).

strTo – The TestDirector names or e-mails of the people that will appear in To field of the e-mail (the names and e-mails should be separated by semicolon. For example: "user@domain.com;admin;alice_td").

strCc – The TestDirector names or e-mails of the people that will appear in Cc field of the e-mail (the names and e-mails should be separated by semicolon. For example: "user@domain.com;admin;alice_td"). Specify an empty string ("") to omit this parameter.

strSubject – The e-mail Subject. Specify an empty string ("") to omit this parameter.

strComment – The e-mail comment (will appear at the top of the e-mail). Specify an empty string ("") to omit this parameter.

Notes:

1. The object ID value can always be retrieved using the {Object}_Fields collection, by retrieving the Value property of the ID field. The example below shows how to retrieve the Ids of Defect, Test and Requirement:

```
            ' Defect ID – for SendDefect
            Bug_Fields("BG_BUG_ID").Value
            ' Test ID – for SendTest
            Test_Fields("TS_TEST_ID").Value
            ' Requirement ID – for SendReq
            Req_Fields("RQ_REQ_ID").Value
```

2. The templates for Defect, Test and Requirement are provided. However the function may be used to work with any object that has Mail function.

3. The third parameter in Mail function allows specifying e-mail options. In the templates below this parameter is hard-coded (its value 2 means that the object History will be sent). However you can change this value to any of the values of TDMAIL_FLAGS. You can use sum to combine the mail properties. For example: 1 – means "send attachments"; 2 – means "send history". In order to send the mail with attachments and history, specify 3 for this argument.

Return Value: None

Code Template:

```
            Sub SendDefect (iObjectId, strTo, strCc,
   strSubject, strComment)
On Error Resume Next
Dim objBugFactory, objBug
Set objBugFactory = TDConnection.BugFactory
Set objBug = objBugFactory.Item(iObjectId)
objBug.Mail strTo, strCc, 2, strSubject, strComment
Set objBug = Nothing
Set objBugFactory = Nothing
            PrintError "SendDefect"
On Error GoTo 0
            End Sub
```

Template 3.5.a

```
            Sub SendTest (iObjectId, strTo, strCc,
    strSubject, strComment)
On Error Resume Next
Dim objTestFactory, objTest
Set objTestFactory = TDConnection. TestFactory
Set objTest = objTestFactory.Item(iObjectId)
objTest.Mail strTo, strCc, 2, strSubject, strComment
Set objTest = Nothing
Set objTestFactory = Nothing
             PrintError “SendTest”
On Error GoTo 0
             End Sub
```

Template 3.5.b

```
             Sub SendRequirement (iObjectId, strTo, strCc,
    strSubject, strComment)
On Error Resume Next
Dim objReqFactory, objReq
Set objReqFactory = TDConnection.ReqFactory
Set objReq = objReqFactory.Item(iObjectId)
objReq.Mail strTo, strCc, 2, strSubject, strComment
Set objReq = Nothing
Set objReqFactory = Nothing
             PrintError “SendRequirement”
On Error GoTo 0
             End Sub
```

Template 3.5.c

Usage Example: In the following example when the test status is changed, the e-mail notification is sent to the designer of the test, cc to all the members of the [QA Testers] group.

```
Sub TestPlan_Test_FieldChange(FieldName)
On Error Resume Next
Dim strSubject, strComment
If FieldName = "TS_STATUS" Then
            strSubject = “Test Change Notification” & _
                        “ for project “ & TDConnection.ProjectName & _
                        “ in domain “ & TDConnection.DomainName
            strComment = “The user “ & User.FullName & _
 “ changed the status of the test ” & _
 Test_Fields(“TS_NAME”).Value & _
 “ to “ & Test_Fields(“TS_ STATUS”).Value
            SendTest Test_Fields("TS_TEST_ID").Value, _
            Test_Fields(“TS_RESPONSIBLE”).Value, “[QA Testers]”, _
                        strSubject, StrComment
End If
On Error GoTo 0
End Sub
```

3.6 Obtaining Defect Statistics for the User

Purpose: The template allows obtaining statistics for the user, for example: how many defects are assigned to current user. The template may be used to create the button that shows this statistics whenever user clicks on it. Another way to use this template is to create the conditions that allow the team leader to decide on tasks. For example: the team leader doesn’t want to allow more then 5 defects with priority Urgent and with status Open or Reopen to be assigned to one person.

Note: The template below is for the Defects. However such function can be created for any object in TestDirector.

Code Location: The code for the each object should be placed in the script of the corresponding module. Each logical set should be placed in separate function as well.

Templates Used: templates 2.1.1, 2.1.2

Arguments:

strFilterConditions – The conditions for which the number of objects should be found. Should be specified in the format
“FILTER_FIELD2=FilterCondition1; FILTER_FIELD2=FilterCondition2;”

While filter conditions format is the same as the format that is used in OTAClient for filters.
For example: if you want to get the number of defects assigned to the user “alex_td”, that have status “Open” or “Reopen” and the priority “4-Urgent”, the following string should be specified: “BG_RESPONSIBLE=alex_td;BG_STATUS=Open Or Reopen;BG_PRIORITY=4-Urgent”
Return Value: Returns the number of objects for given condition

Code Template:

```
Function GetDefectStatistics(strFilterConditions)
On Error Resume Next
Dim strFilterConditionArr, strCondition
Dim objBugFactory, objFilter, objList
Set objBugFactory = TDConnection.BugFactory
Set objFilter = objBugFactory.Filter
strFilterConditionArr = Split(strFilterConditions, “;”)
For i = 0 To Ubound(strFilterConditionArr)
 strCondition = Split(strFilterConditionArr(i), “=”)
If Ubound(strCondition) = 1 Then
        objFilter.Filter(strCondition(0)) = strCondition(1)
End If
Next
Set objList = objFilter.NewList
GetDefectStatistics = objList.Count
Set objList = Nothing
Set objBugFactory = Nothing
Set objFilter = Nothing
        PrintError “GetDefectStatistics”
On Error GoTo 0
End Function
```

Template 3.6

Usage Example: In the following example when the defect with priority “4-Urgent” is assigned to some user, workflow checks how many defects with the same priority and the status “Open” or “Reopen” are already assigned to the same user. If the number of defects is 5 or more, the workflow will show warning.

```
Sub Defects_Bug_FieldChange(FieldName)
On Error Resume Next
Dim strConditions, iNumberOfBugs
If FieldName = "BG_RESPONSIBLE" Then
        strConditions = “BG_RESPONSIBLE=” & _
                                Bug_Fields(“BG_RESPONSIBLE”).Value & _
                                “;BG_STATUS=Open Or Reopen” & _
                                “;BG_PRIORITY=4-Urgent”
        iNumberOfBugs = Cint(GetDefectStatistics(strConditions))
        If iNumberOfBugs >= 5 Then
                MsgBox “” & iNumberOfBugs & “ defects” & _
                                “ with Priority=<4-Urgent> and “ & _
                         “ Status=<Open> or <Reopen>” & _
                                “ are already assigned to the user “ & _
                          Bug_Fields(“BG_RESPONSIBLE”).Value
        End If
End If
On Error GoTo 0
End Sub
```

3.7 Setting the Last Item in List As a Default Value of the Field

Purpose: The template allows to select the last item from the list, and to set it as a default value for some field. This is useful for the fields where the last value is most commonly used. For example: the field Detected In Version has a list of versions attached. Most of the users will select the current version for the defects they submit, which is the last in the list. So this value can be a default for this field.

Code Location: code should be added to the code of each workflow script (Defects, Test Plan, etc.). Separate procedures should be created for each object for which this functionality is needed. The {Object} statement should be replaced with the name of the object for which the template is used (for example: Bug, Test, etc.).
Templates Used: templates 2.1.1, 2.1.2
Arguments:
strFieldName – The name of the field for which the default value should be set.
strListName – The name of the list from which the default value should be selected. This list should be attached to the field specified by *strFieldName*.
Return Value: None

Code Template:

```
Sub SetDefaultValue(strFieldName, strListName)
On Error Resume Next
    Dim objCustomization, objLists, objList, objNode, objChildren
    Dim iChildrenCount
    Set objCustomization = TDConnection.Customization
    objCustomization.Load
    Set objLists = objCustomization.Lists
    Set objList= objLists.List(strListName)
    Set objNode = objList.RootNode
    iChildrenCount = objNode.ChildrenCount
    Set objChildren = objNode.Children
    {Object}_Fields(strFieldName).Value = objChildren(iChildrenCount).Name
    objCustomization.Commit
    Set objCustomization = Nothing
    Set objLists = Nothing
    Set objList= Nothing
    Set objNode = Nothing
    PrintError "SetDefaultValue"
On Error GoTo 0
End Sub
```

Template 3.7

Usage Example: In the following example when the New Defect form is opened, the script assigns "Versions" list to Detected in Version (BG_DETECTION VERSION) field. Then the value of the Detected In Version field is set to the last item in Versions list.

```
Sub Defects_Bug_New()
On Error Resume Next
    Bug_Fields("BG_DETECTION_VERSION").List = Lists("Versions")
    SetDefaultValue("BG_DETECTION_VERSION", "Versions")
On Error GoTo 0
End Sub
```

3.8 Copy Last Run Value to Test in Test Set

Purpose: The template allows copying the value from last test run to test in test set. For example: each time the test is executed, the user wants the value of Duration field of the run to be copied to the Expected Duration user-defined field of test in test set.
Code Location: code should be added to the code of Test Lab workflow script
Templates Used: templates 2.1.1, 2.1.2
Arguments:
strSrcRunField – The name of the Run field from which the value should be copied.
strTrgTSTestField – The name of the Test in Test Set field to which the value should be copied.
Return Value: None

Code Template:

```
Sub CopyLastRunValue(strSrcRunField, strTrgTSTestField)
On Error Resume Next
    Dim iTestSetId, iTSTestId, objTestSet, objTSTest, objLastRun
      iTestSetId = TestSetTest_Fields("TC_CYCLE_ID").Value
```

```
            iTSTestID = TestSetTest_Fields("TC_TEST_ID").Value
  Set objTestSet = TDConnection.TestSetFactory.Item(iTestSetId)
            Set objTSTest = objTestSet.TSTestFactory.Item(iTSTestID)
            Set objLastRun = objTSTest.LastRun
  If objLastRun.Field(strSrcRunField) <> TestSetTest_Fields(strTrgTSTestField).Value Then
          TestSetTest_Fields(strTrgTSTestField).Value = LastRunObject.Field(strSrcRunField)
            End If
  Set objTestSet = Nothing
            Set objTSTest = Nothing
            Set objLastRun = Nothing
          PrintError “CopyLastRunValue”
On Error GoTo 0
End Sub
```

Template 3.8

Usage Example: In the following example the user has Expected Duration field for Test in Test Set (TC_USER_01). The value of this field is updated any time the value of Duration field (RN_DURATION) of the last run differs from Expected Duration of the test.

```
Sub TestLab_TestSetTests_MoveTo
On Error Resume Next
    CopyLastRunValue "TC_USER_01", "RN_DURATION"
On Error GoTo 0
End Sub
```