Test Automation Approaches to Real Life

Yury Makedonov, CGI

Golam Mustofa, CGI

The International Quality Conference, Toronto, Ontario, Canada

October 1, 2003

Agenda

- **Why Automate?**
- **Reality of GUI Test Automation today**
- **What are the challenges?**
- **Test Automation Approaches in CGI**
  - **Stand Alone Automation Approach**
  - **Fully Integrated Automation approach**
- **Summary**
- **Q&A**

Primary Uses Of GUI Test Automation

- **Perform regression tests**
- **Volume tests**
- **Reduce human resource requirements**

Why Automate?

- **Save Time**
- **Reduce Total $$ (COQ)**
- **Repeatability/More Consistent Coverage**
- **Reliability**
- **Reusability**
- **Portability**

Reality Of GUI Test Automation Today

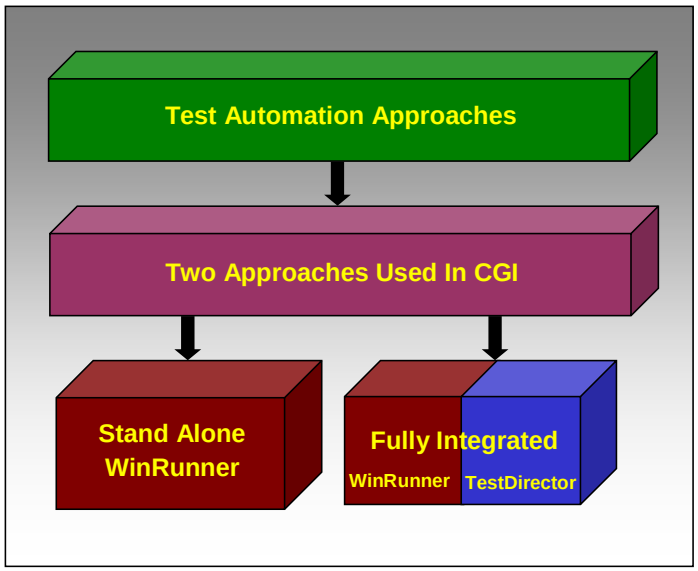
By many accounts:

- **More than 50% of GUI Test Automation projects fail**
- **Test scripts & tools become shelfware**

What Are The Challenges?

- **GUIs are dynamically changing**
- **Application is not stable enough (last minutes changes, scope creep)**
- **Not enough time to update scripts (poorly designed)**
- **Lack of in-house expertise**
- **Budget constraints**
- **Immature Test Automation methodologies**

Test Automation Approaches in CGI



Standalone WinRunner

Yury Makedonov

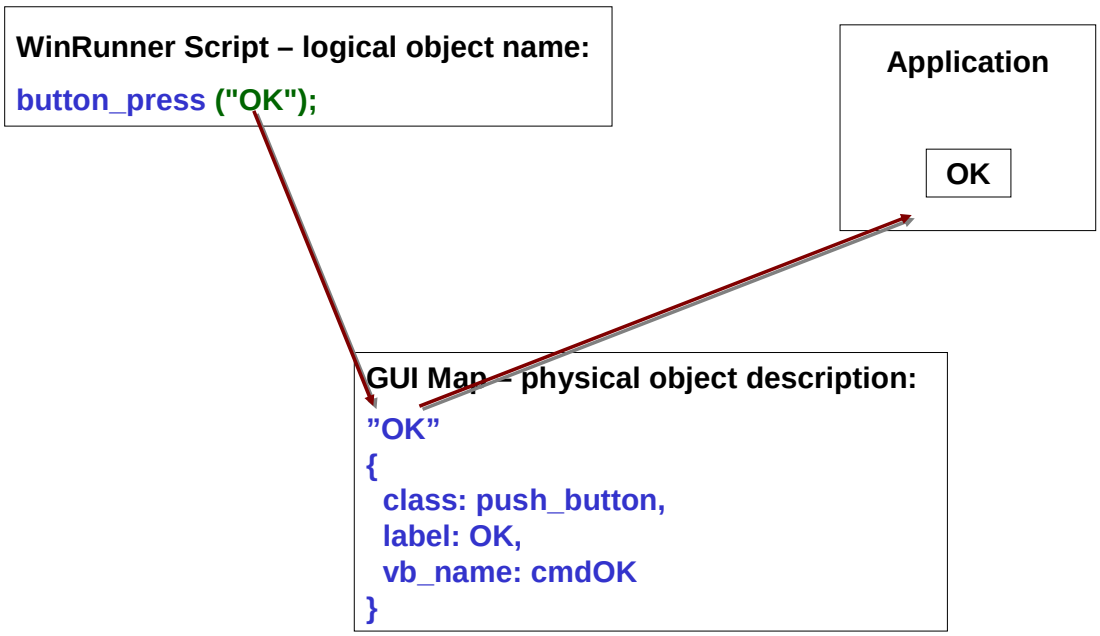
Standalone WinRunner

The fundamental principles will be covered in this section

Test automation from point of view of:

- **Portability of test scripts**
- **Maintainability of test scripts**
- **Test Automation team supporting**:
  - Multiple versions
  - Of multiple applications
  - In different environments
  - In different locations

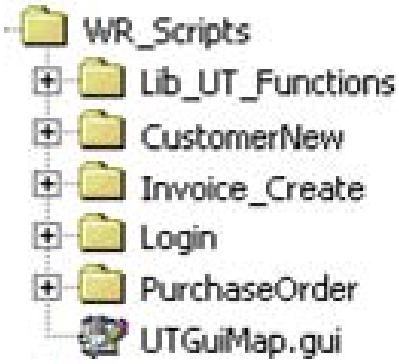
What is a GUI Map?



Step One – Tools evaluation

- **Can a GUI testing tool handle our application?**
- **How time consuming is test automation (Does it make sense to automate)?**
- **Do we care about a proper organization of test scripts?**

Step Two – First version of the application:

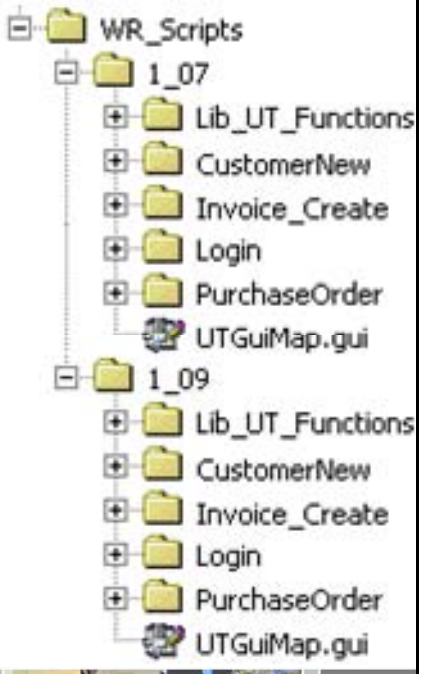



```
load("C:\\WR_Scripts\\Lib_UT_Functions");
GUI_load("C:\\WR_Scripts\\ UTGuiMap.gui");
web_browser_invoke(IE,"http://confut.xyz.ont.ca");
UT_Select_Language("English");
UT_Login("a327012", "12345");
```

Step Three – Second version:

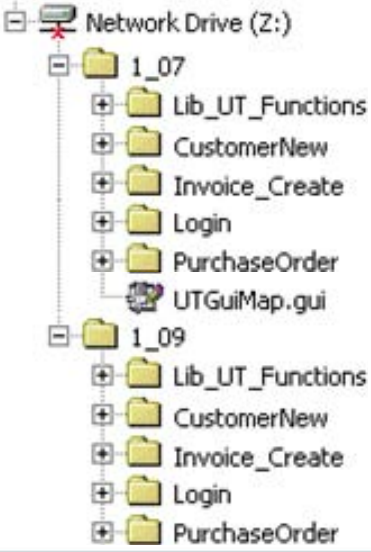
- New version of the application delivered
- Cloning of scripts
- No full path for a GUI map or a script

```
load("Lib_UT_Functions");
GUI_load("UTGuiMap.gui");
```



Step Four – Several Testers:

- Several testers working with WinRunner
- Scripts moved onto shared Z: drive



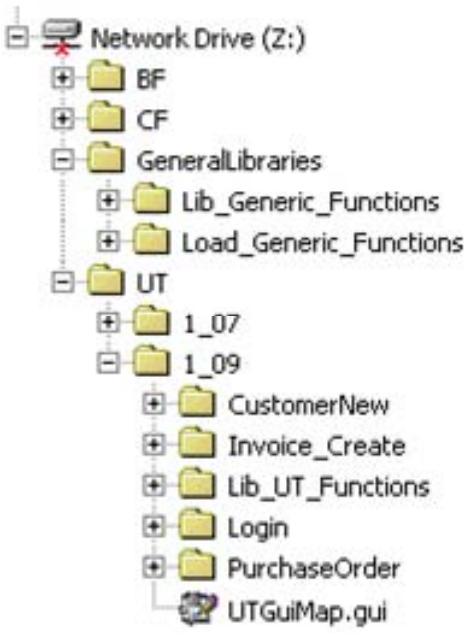

```
load("Lib_UT_Functions");

GUI_load("CoolGuiMap.gui");

UT_Open_Site( );

UT_Select_Language("English");
UT_Login("a327012", "12345");
UT_Select_Portfolio("KEY");

call "Invoice_Create"( );
```

Step Four – Several Testers:

- Simultaneous execution/modification of the same script/library on different workstations.
- All modifications are performed on a local copy of a script.
- After debugging script will be copied onto a shared drive.
- Each script has an “owner”. Owner stores “master” copy of the script on his local workstation.

Step Five – Several Applications:

- Test automation for several applications
- Application (version) specific libraries
- General libraries (used by all projects)
- Major rework of libraries of functions



Step Six – Several Environments, Browsers, User Roles:

We need to execute the same script concurrently:

- against different environments,
- using different browsers,
- using different user roles/login IDs.

- Store setup parameters in a local file.

- C:\TestData\UTEnvironment.xls file:

URL	Browser	UserID	Password
http://prodaceman1.configurator.com/ut	IE	a327842	123

```
Site_Open(flURL, flBrowser, fluserID, flPassword);
```


Recommendations for WinRunner projects of any complexity:

- Implement the complete hierarchy of folders for WinRunner scripts from day one:
  - Folder for general libraries
  - Folders for different Application
    - ➔Subfolders for different Versions of the application
    - ➔All scripts for a specific version of an application are stored on the same level of hierarchy
- Use path relative to a current test to load an object (No specification of a full path, starting with a drive letter, allowed)
- Implement a standard script header containing description of a script + all initialization steps:
  - GUI Load,
  - Libraries load,
  - Reading data file, etc.
- Implement “Script Ownership” concept: ***master copy of any script belongs to a single person***

MAKE IT A STANDARD!!!

Fully Integrated Test Automation - WinRunner & TestDirector

Golam Mustofa

Step One – Know The Tools

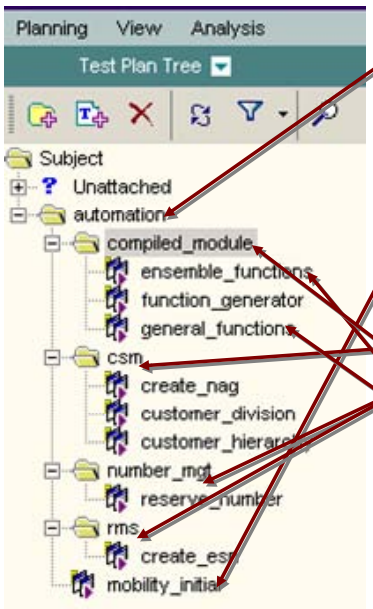
- **Functional Testing Tool**
- **Develops the scripts based on detail Test Cases**
- **Needs to create GUI file and Data Spreadsheet**
- **Needs to connect to the TestDirector**

- **Test Management Tool**
- **WinRunner scripts have to be integrated with TestDirector**
- **TD schedules execution of WinRunner scripts**
- **Manages Defects**

Step Two – Understand The Overall Process

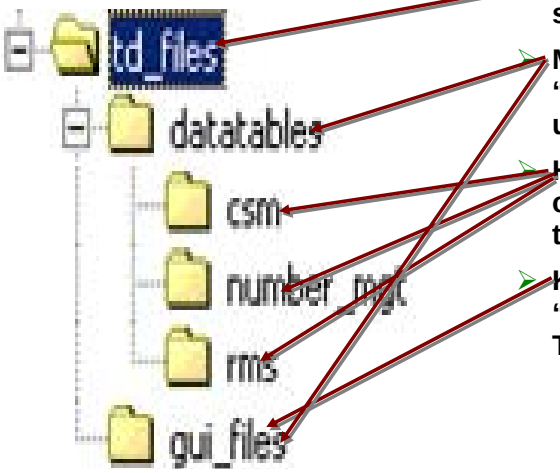
- **WinRunner activities…**
  - **Learn GUIs of the application under testing**
  - **Develop / Record scripts**
  - **Debug scripts**
  - **Import or Save scripts into the TestDirector**
- **TestDirector activities…**
  - **Locate the Test Data Spreadsheet into a shared drive or attach to the TestDirector**
  - **Build the Test Sets**
  - **Schedule execution of the Test Sets**
  - **Analyse Test Result and create Test Report**
  - **Report or Manage defects**

Step Three – Create An Effective Framework



- **Create a root folder into the Test Plan Tab of TD**
- **Design an Initial / Startup scripts…**
  - ✓ **Test Environment setup**
  - ✓ **Load GUI and function library**
  - ✓ **Declare variables, data spreadsheet location, etc**
  - ✓ **It itself can have a data spreadsheet as well**
  - ✓ **Example…**
- **Create sub folders based on components of the AUT**
- **Make a folder for compiled module or functions library…**
  - ✓ **Function library for generic functions**
  - ✓ **Function library for application specific functions**

Step Three – Create An Effective Framework...



- Build a "td_files" folder into the shared drive
- Make sub folders named "datatables" and "gui_files" under the "td_files" folder
- Have sub folders based on components of the AUT under the "datatables" folder
- Keep all the GUI files into the "gui_files" folder or TestDirector's project

Step Four – Utilize Data-Driven Test Techniques

- Single scripts will handle negative and positive boundary test reading data from the data spreadsheet

	run_more	business_name	test_result
1	X	Bell Canada	Passed
2		Bell Mobility	Passed
3	X	Bell ExpressVu	Passed
4		BCE Media	Passed
5	X	Bell Emergis	Passed
6		BCE Nexxia	Passed
7	Stop	BellZinc.ca	Passed
8		Sympatico	Passed
9		BCE ActiMedia	Passed
10		$!% #	Failed

Blank data

Special characters

Valid data

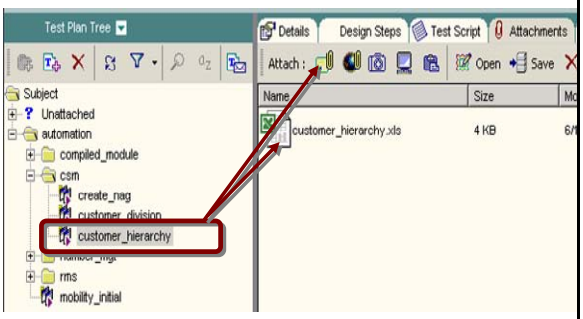
Error Message

Accepted

- Have control on the spreadsheet…
  - Skip some rows
  - Stop in row
- Write important data or results into the spreadsheet after the transactions for future analysis

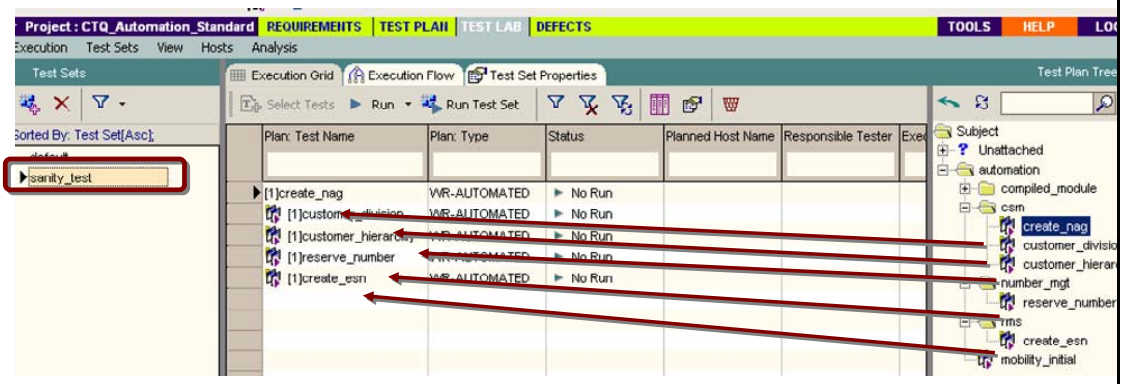
Step Five – Locate Data Table (Spreadsheet)

- **TestDirector doesn't allow to save the Data spreadsheet into the Project's database …**
  - **Attach the data spreadsheet with the Test Case**
  - **Download the spreadsheet on the fly during scripts execution**
  - **Read the spreadsheet as input file**
  - **Or keep the spreadsheet into the shared drive**



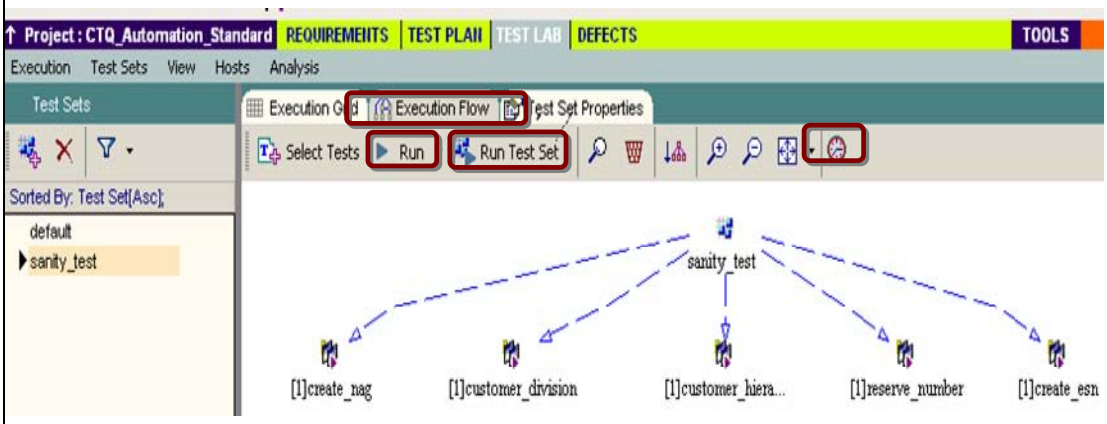
Step Six – Build Test Sets

- **Identify all Testing scripts which needs to be into a bundle based on application's functionality**
- **Create a Test Set**
- **Attach all identified Testing scripts into the Test set**



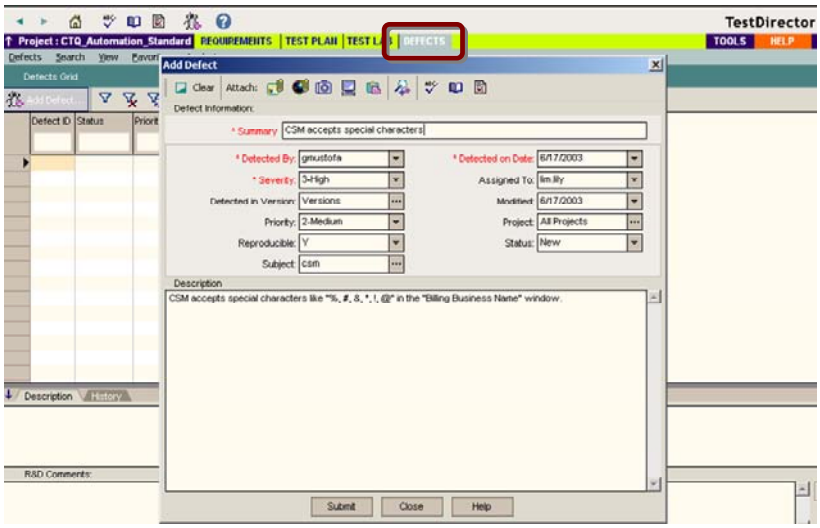
Step Seven – Schedule And Execute Test Sets

- From 'Execution Flow" tab click on clock icon
- Set the time and date you'd like to the Test Set or Individual Test
- Press on "Run Test Set" or "Run" button
- TestDirector will invoke WinRunner and perform all the tests



Step Eight – Manage Your Defects

- Analyse the Test Result
- Manage any defect through the "Defect" Tab of TestDirector



Summary

- **Test Automation, when used to perform Regression & Volume testing can reduce cost, human resources and time**
- **Implement Stand Alone or Fully Integrated method based on your environment, goals/priorities, budget and expertise of resources**
- **Test Automation is worth the investment and effort, if it's implemented properly!!!!**

Questions??????

Appendix A – Additional Material

Yury Makedonov

Appendix – Standalone WinRunner:

How to load an object from “General Libraries” folder (GUI map, library of functions, data file) from a main test script:

- Use path relative to a parent script `PathOneLevelUp()` function
- Load all common libraries during WinRunner initialization,
- Use global variable to store path to this folder.

Appendix – Standalone WinRunner:

How to load a general library from an “Application” script, without specifying a drive letter explicitly, using PathOneLevelUp() function :

```
load(PathOneLevelUp(PathOneLevelUp(PathOneLevelUp
(getvar("testname")))) & "General libraries"
& "\\" & "Lib_Generic_Functions");
```

Sample code of PathOneLevelUp() function:

```
function PathOneLevelUp( varPath )
{
 auto numElements, path_array[], varPathOneLevelUp = "", i;
 numElements = split (varPath, path_array, "\\");
 delete path_array[numElements];
 for (i=1; i<numElements; i++)
   varPathOneLevelUp = varPathOneLevelUp & path_array[i] & "\\";
 return varPathOneLevelUp;
}
```


Appendix B – Additional Material – Sample Scripts

Golam Mustofa

Appendix B - Designing Initial / Startup scripts...

```
##############################################################################################
table = get_uid=create_input_dialog ("Please type in your UserId");
rc = ddt_open(table, DDT_MODE_READ);
if (rc!= E_OK && rc != E_FILE_OPEN)
    pause("Cannot open table.");
ddt_get_row_count(table,table_RowCount);
for(table_Row = 1; table_Row <= table_RowCount; table_Row ++)
{
    ddt_set_row(table,table_Row);
    test_path = ddt_val(table, "test_path");              # Share Drive where Datatable and GUI files are located
    td_path = getvar("testname");
    td_path = substr(td_path, 1, index(td_path, ddt_val(table, "initial_name"))); # Get the root directory
    ############## GUI Definition ###################################################################
    GUI_close_all();                                      #Close all existing GUI Map files
    GUI_load(gui_path & ddt_val(table, "initial_name"));  # Load CSM.gui GUI Map files
}
ddt_close(table);
    ############ Const ##############################################################
    public test_path;
    public const data_path = test_path & "DataTables\\";    # Get the Datatable path from share drive
    public const gui_path = test_path & "GuiFiles\\";       # Get the GUIFiles path from share drive
    public const function_path = td_path & "General Tests\\Compiled Modules\\";
     # Get the Function_Path path from TestDirector
    ######################## Function ###############################################################

    reload(function_path & "gnrc_fncs",0,1);                          # Reload the Generic functions
    reload(function_path & "esbl_fncs",0,1);                          # Reload the Ensemble Functions
    m_root = getenv("M_ROOT");
    if (m_root != "")
      (reload(m_root & "\\lib\\win32api",1,1);)                       # Load Win32api function
```


Appendix B - Skip some rows or in a row on the data spreadsheet

```
table = "C:\\Demo\\create_new_account\\create_new_account.xls"; #Open Datatable for Consumer BAN
rc = ddt_open(table, DDT_MODE_READWRITE);
if (rc!= E_OK && rc != E_FILE_OPEN)
    pause("Cannot open table.");
ddt_get_row_count(table,table_RowCount);
for(table_Row = 1; table_Row <= table_RowCount; table_Row ++)
{
    ddt_set_row(table,table_Row);                         # Focus on the 1st row of csm_crte_csbn_0001 Datatable
        need_more=tolower(ddt_val(table, "run_more"));
    if (need_more =="stop")
        {
            report_msg("Test was stopped at: "&table_Row"th row of your datatable as you had a break point.");
            texit;
        }
    if (need_more =="x")
        (ddt_set_row(cust_hrch,cust_hrch_Row);)           # Go to the next row of the datatable
    if (need_more!="x")
        {
    #|-------------- Script Starts from here ------------------------------------|
#   ...
#   ...
        }
}
```

Appendix B - Locate Data Table (Spreadsheet)

```
# How do I save a Datatable in TestDirector and access it during the script run time thru TD?
# Solutions:
#You can' t save the Datatable in TD, but there are two workarounds.
# 1. Attach the Datatable as a file in Test Case on TestDirector (under Test Plan Tab) and use the code below in WR.

    if (getvar("td_connection") == "on")
# It's assumed that you are running the script thru TD, if you're currently connected to TD.
        {
            myDatatable =tddb_load_attachment ("Sanity Test.xls", test_path );
            # "Sanity Test.xls" is the name of the Datatable as attached file in TD
            ddt_open(myDatatable, DDT_MODE_READ);
        }
    else
        {
             myDatatable = "\\\\myserver\\sharefolder\\ Sanity Test.xls";
             # Else look at the shared Drive or Directory
             rc = ddt_open(myDatatable, DDT_MODE_READ);

        }

#2. Keep your Datatable in a shared drive and access it thru the Network and modify your something like this...


    myDatatable = "\\\\myserver\\sharefolder\\Sanity Test.xls";
    rc = ddt_open(myDatatable, DDT_MODE_READ);
# In this example, you have a mapped drive to the "sharefolder" folder in the machine called "myserver"
# and the "Sanity Test.xls" file is stored there.
```


Contact Information

Yury Makedonov

Senior Consultant,
Centre for Testing & Quality
CGI Services to BCE
595 Bay Street, Floor 10-B
Toronto, Ontario,
Canada M5G 2C2
(416) 215-3593
yury.makedonov@cgi.com

Golam Mustofa

Senior Consultant,
Centre for Testing & Quality
CGI Services to BCE
595 Bay Street, Floor 10-B
Toronto, Ontario,
Canada M5G 2C2
(416) 215-7534
golam.mustofa@cgi.com