Evolution of a WinRunner framework from a pilot project to a multi-project test team (on the example of a web application)

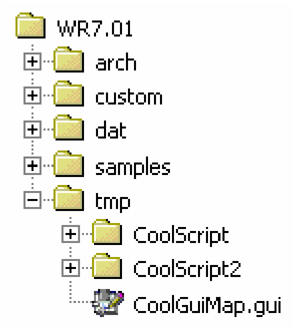
I discuss here the evolution of organization of WinRunner scripts (WinRunner framework) through the different steps on the example of a web application. It would be presented as a story of a fictious successful company that starts with their first WinRunner tester and grows up to test automation team supporting several projects.

For every step I discuss the possible problems and their solutions.

Beginning:

First WinRunner scripts developer + first version of the application.

All WinRunner scripts are stored on C: drive.



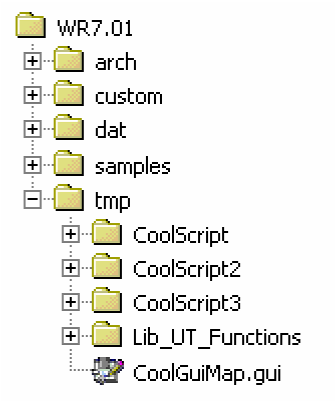
```
# XYZ Inc.
# Test Automation Team
# Cool WinRunner script by Yury M.
# Description: Tests the application

GUI_load("C:\\Program Files\\Mercury\\WinRunner\\tmp\\CoolGuiMap.gui");
web_browser_invoke(IE,"http://confut.bell.ont.ca");

Blah();
Blah();
```

Step 2 – libraries of functions

Introduction of libraries of functions.



```
# XYZ Inc.
# Test Automation Team
# One more cool WinRunner script by Yury M.
# Description: Thoroughly tests the application

load("C:\\Program Files\\Mercury\\WinRunner\\tmp\\Lib_UT_Functions");

GUI_load("C:\\Program Files\\Mercury\\WinRunner\\tmp\\CoolGuiMap.gui");
web_browser_invoke(IE,"http://confut.bell.ont.ca");

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

Step 3 – different environments: QA, Mirror, Production

We have to run the same script against different environments: QA, Mirror, Production, etc.
Hardcoded `UT_Site_Open()` function.

Main Test:

```
Load("C:\\Program Files\\Mercury\\WinRunner\\tmp\\Lib_UT_Functions");

GUI_load("C:\\Program Files\\Mercury\\WinRunner\\tmp\\CoolGuiMap.gui");

UT_Site_Open( );

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

Compiled Module

```
#=========================================================================
function UT_Site_Open( )
# Version #     Created by      Date              Version
# 1.0           ymakedonov      Mar 11, 2002     Initial version
#-------------------------------------------------------------------------
#Opens a new UT site in a new browser window:
{
        auto autoUTSiteURL =  "http://confut.bell.ont.ca";

        report_msg ( "Will try to open site '" & autoUTSiteURL & "'");
        web_browser_invoke ( IE, autoUTSiteURL );
        return 0;
}
```

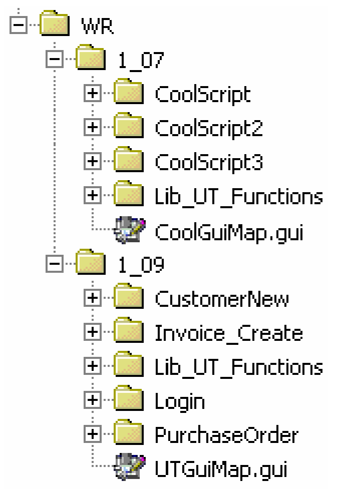
Step 4 – New version of the application delivered

New version of the application delivered – we have different versions in QA environment and production – and have to support them simultaneously.

Cloning of scripts

To start development of test scripts for new version of the application we copy the folder with all scripts and other files and use the *Application version* as a name of this folder. Also we rename the original folder.

To use this approach we should not specify a full path for a GUI map or a script. Only names are allowed!



```
WR
  1_07
    CoolScript
    CoolScript2
    CoolScript3
    Lib_UT_Functions
    CoolGuiMap.gui
  1_09
    CustomerNew
    Invoice_Create
    Lib_UT_Functions
    Login
    PurchaseOrder
    UTGuiMap.gui
```

```
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 5 – new member of a test automation team

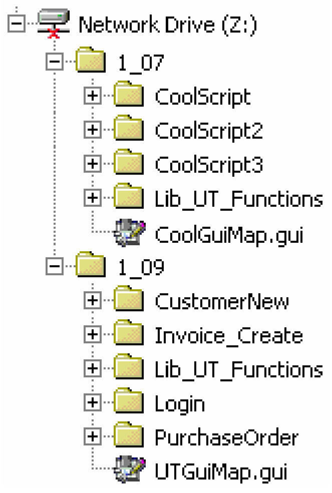
Second (junior) tester working with WinRunner appeared. Only scripts execution. Master – Apprentice relationship.

Everything is stored on C: drives.

Apprentice only copies scripts developed by a senior tester and executes them.

Step 6 – more members of a team

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



```
Network Drive (Z:)
  1_07
    CoolScript
    CoolScript2
    CoolScript3
    Lib_UT_Functions
    CoolGuiMap.gui
  1_09
    CustomerNew
    Invoice_Create
    Lib_UT_Functions
    Login
    PurchaseOrder
    UTGuiMap.gui
```

```
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 7 – concurrent access for modification/execution

Simultaneous execution/modification of the same script/library on different workstations.

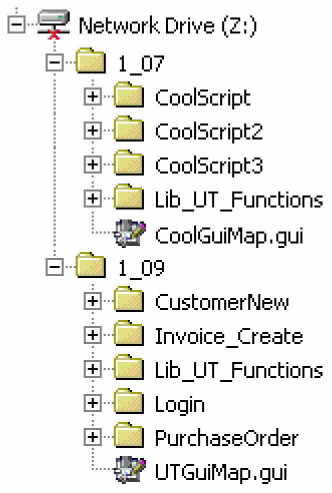
All modifications are performed on a local copy of a script.

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.

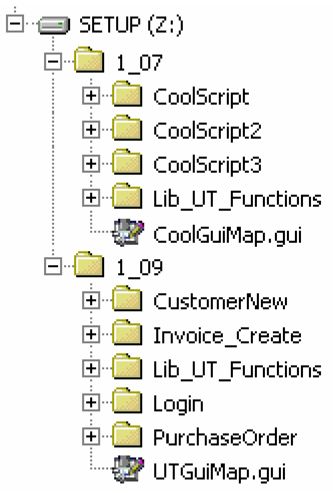
Re-mapping server/workstation using "*subst*" command as an option.

```
subst Z: Y:\WRscripts
```



```
Network Drive (Z:)
  1_07
    CoolScript
    CoolScript2
    CoolScript3
    Lib_UT_Functions
    CoolGuiMap.gui
  1_09
    CustomerNew
    Invoice_Create
    Lib_UT_Functions
    Login
    PurchaseOrder
    UTGuiMap.gui
```

```
subst Z: C:\WRscripts
```

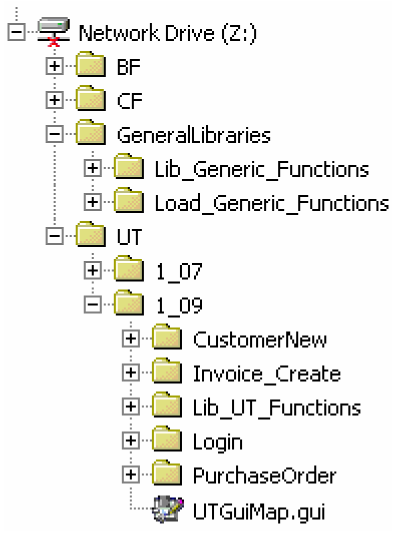


```
SETUP (Z:)
  1_07
    CoolScript
    CoolScript2
    CoolScript3
    Lib_UT_Functions
    CoolGuiMap.gui
  1_09
    CustomerNew
    Invoice_Create
    Lib_UT_Functions
    Login
    PurchaseOrder
    UTGuiMap.gui
```

Step 8 – test automation for several applications

Team got several applications to perform test automation.

Application (version) specific libraries and general libraries (used by all projects) are stored in different folders.



How to load an object from “General Libraries” folder (GUI map, compiled module, data file) from a main test script?

Just remember the rule “No full path”!

Solution A:

Use `PathOneLevelUp()` function and *“testname”* variable to find an object stored in another folder. We can use this function recursively. Path relative to a launched script.

Example showing how to load a general library from an “Application” script:

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

Code of PathOneLevelUp() function:

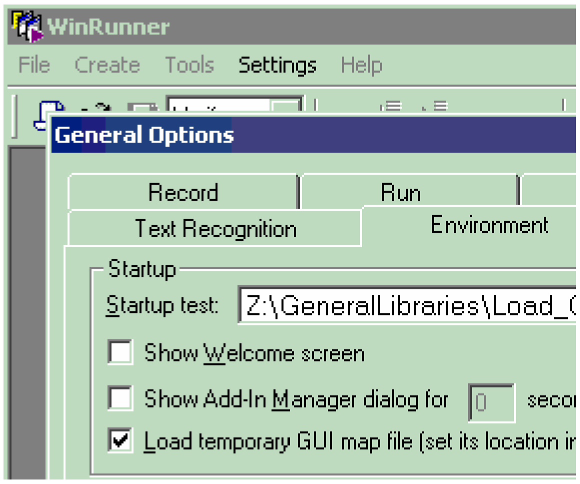
```
function PathOneLevelUp( varPath )
# Version #     Created by      Date                Version
# 1.0           ymakedonov      Aug 23, 2001        Initial version
# This functions returns a path of a parent folder
{
      auto numElements, path_array[], varPathOneLevelUp = "", i;

      numElements = split (varPath, path_array, "\\");
      delete path_array[numElements];
      for (i=1; i<numElements; i++)
            varPathOneLevelUp = varPathOneLevelUp & path_array[i] & "\\";

      return varPathOneLevelUp;
}
```

Solution B:

Use a “Startup Test” to load generic libraries.

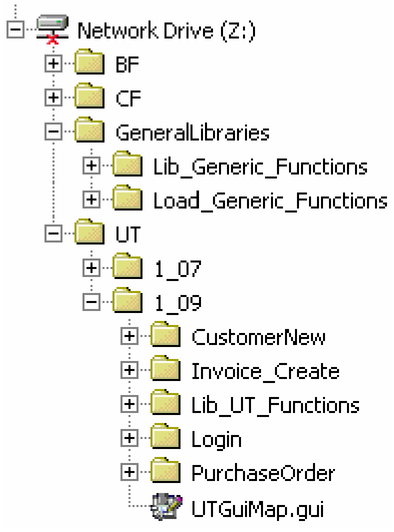


Startup test: Z:\GeneralLibraries\Load_(

☐ Show Welcome screen

☐ Show Add-In Manager dialog for 0 secor

☑ Load temporary GUI map file (set its location ir



Main goal of Load_Generic_Functions() is to load a common library:

```
function Load_Generic_Functions()
# Version #     Created by      Date            Version
# 1.0           ymakedonov      Aug 23, 2001    Initial version
# Loading a generic library
{
        load("Lib_Generic_Functions",0,0);
}
```

This function can also be used to perform other initialization tasks.

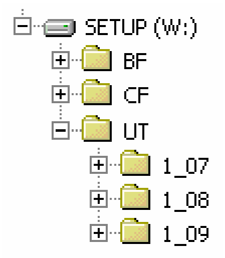
Step 9:

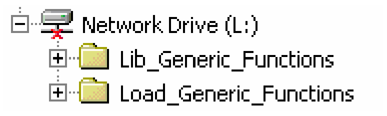
To decrease risk and streamline the logic we can store general libraries and scripts for a particular application on a different drive letters.

Only application scripts would be modified frequently. People won’t be updating general libraries very often.

We can move script onto different drive letters (C:, X:, Y:, Z:, etc.) without any code modification due to the “No full path” rule.

```
subst W: C:\WRscripts               subst L: Y:\Wrlibraries
```





Step 10:

We need to execute the same script concurrently:

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

Storing setup parameters in a file on a local workstation.
C:\TestData\UTEnvironment.xls file:

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

Fragment of a script retrieving these parameters:

```
vLocalTestEnvironmentXls = "C:\\TestData\\UTTestEnvironment.xls";

# Retrieve environment variables from a local EXCEL file:
ddt_open(vLocalTestEnvironmentXls);
vURL = ddt_val (vLocalTestEnvironmentXls, "URL");
vBrowser = ddt_val (vLocalTestEnvironmentXls, "Browser");
vUserID = ddt_val (vLocalTestEnvironmentXls, "UserID");
vPassword = ddt_val (vLocalTestEnvironmentXls, "Password");
ddt_close(vLocalTestEnvironmentXls);
```

To make a decision

- To rely or not to rely on *wrun.ini* file?
  wrun.ini files should be synchronized on all machines.
- To use or not to use CSO library?
  Not a lot of functions. Conflict between CSO functions and in house developed functions.
- Exception handling?
  Conflicts of generic and project specific exceptions.

Absolute portability

How to develop scripts that could be copied between file system and TestDirector?

```
rc = getvar ("td_connection");
report_msg ("td_connection = " & rc );

if (rc == "on")      {
       #Loading GUI map and libraries from TestDirector database:
       rc = GUI_load("TD_UT_GUI.gui");
       load("[TD]\\Subject\\ZTest\\Lib_UT_Functions",0,0);
       }
else   {
       #Loading GUI map and libraries from a file system:
       rc = GUI_load("UT_01.gui");
       load("Lib_UT_Functions",0,0);
       }
```

Recommendations for project of any complexity

I recommend the following measures regardless of a complexity of your test automation projects.
In a long run they will save a lot of time for you.

Implement the complete hierarchy of folders for WinRunner scripts from day one:

- Folder for general libraries,
- Folder for **All Application Scripts**
  - Subfolders for different **Applications**
    - Subfolders for different **Versions** of the application
    - All scripts for a specific version of an application are stored on the same level of hierarchy. Subfolders to group scripts are not allowed (one more level of folders hierarchy).

Relative path to all objects, "testname" variable:

No specification of a full path starting with a drive letter.
Use relative path to all objects (scripts, GUI maps, data files) and "testname" variable.

Implement a standard script header containing description of a script + all initialization steps:

- GUI Load,
- Libraries load,
- Reading data file, etc.

Load general libraries using Startup Test:

Startup Test can also be used to:
- Load GUI maps common for all projects,
- Read common data files, etc.

Implement "Script Ownership" concept:

Master copy of any script belongs to a specific person.

Make it a standard.