

ZUG

Day 4

Topics

- Environment sensors
- Concurrent execution
- Zug command lines
- Troubleshooting
- Multi-environment testing
- Maintainability & design

Environment sensors

Environment sensors

- **Zug can automatically sense the environment on which it is running.**
 - **ZEnv_Sensor Molecule**
 - **ZEnv_Values Context Variable**

Environment sensors

- **ZEnv_Sensor** molecule is defined with this name to sense the environment in which the tests are being run
- Users don't call this Molecule explicitly
- Invoked implicitly **after** running the Init test case or after running the first test case of the test suite

Automatic sensors from ZUG

- **ZEnv_Values** context variable is defined by ZUG to store some environment related information.
- Initially ZUG populates it with the values for
 - **OS Name**
 - **Java version**
 - **LOGON SERVER**
 - **COMPUTER NAME**
 - **USER NAME**
 - **APPDATA**
 - **USERDOMAIN**
 - **PROCESSOR_ARCHITECTURE**
 - **NUMBER_OF_PROCESSORS**
 - **RAM SIZE**

Reporting sensors to Zermatt

- Values are stored as comma separated list of key=value pairs, e.g, OS Name=Windows 7,Java Version=1.6,User Name=Ellora
- Test Case designer can use this ZEnv_Values context variables inside the ZEnv_Sensor molecule.
- Can append additional sensors in the ZEnv_Values context variable in ZEnv_Sensor molecule as required e.g, name and version of the browser, build information, etc...
- These sensor values are automatically reported to Zermatt

Example- Environment sensors

Molecule ID	Description	Property	Step	Action	ActionArg_1	ActionArg_2	Verify
zenv_sensor				appendtoContextvar	Zenv_Values	,Browser=IE	BrowserVersion=9
				appendtoContextvar	Zenv_Values	,.NetFrameWork=4	
				SetContextVar	appValues		
				@getApplicationValues	appValues		
				appendtoContextvar	Zenv_Values	,	%appValues%

TC001				print	%ZEnv_Values%
-------	--	--	--	-------	---------------

Running TestCase ID TC001 On(Current Date): 2013:06:26-17:23:17

[TC001] Executed Action print with values [OS Name=Windows 7,Java version=1.7,LOGON SERVER=\\ELLORA,COMPUTER NAME=ELLO
RA,USER NAME=skhan,APPDATA=C:\Users\skhan\AppData\Roaming,USERDOMAIN=ELLORA,PROCESSOR_ARCHITECTURE=x86,NUMBER_OF_PROCE
SSORS=2,RAM=3999 MB,Browser=IE,BrowserVersion=9,.NetFrameWork=4,TestingApplication=MyApp,ApplicationVersion=5.6 U 2]

Concurrent Execution

Concurrency – use cases

- Speed up test case execution
- Introduce complexity, stress and randomness to test application resilience
- Handle unpredictable order of completion
e.g. wait for one of multiple events/outcomes that could happen
- Atom (e.g. a click on a link) that does not return control, until an alert is acknowledged

Concurrent Execution

- Concurrency or parallel execution

- Test Case
- Molecules
- Test Step

Test Case Concurrency

• Test Case

- Concurrency achieved by using MVM or MVCV and GCE property.
- Test Case or Molecule is executed for each combination of the MVM or MVCV values.
- Resulting Test cases or Molecules are then run concurrently (parallel Execution).

Comment:							
Login		GCE		SetContextVar	handle##		
				Zbrowser.Initialize	handle##	\$\$Browser	
				Zbrowser.GoToURL	%handle###%	\$URL	
				Zbrowser.SetTextByName	%handle###%	\$EmailFieldName	\$emailID
				Zbrowser.SetTextByName	%handle###%	\$PasswdFieldName	\$password

Molecule Concurrency

● Molecule

Molecule ID	Description	Property	Step	Action	ActionArg_1	ActionArg_2	ActionArg_3	ActionArg_4	ActionArg_5	ActionArg_6	ActionArg_7
Login		GCE		#define_arg	handle##	Browser	URL	EmailField	PassField	Email	Password
				Zbrowser.Initialize	#handle##	##Browser					
				Zbrowser.GoToURL	%#handle##%	#URL					
				Zbrowser.SetTextByName	%#handle##%	#EmailField	#Email				
				Zbrowser.SetTextByName	%#handle##%	#PassField	#Password				

Notice the usage of ## as a prefix and a suffix.

prefix => interpret the named argument as a MVM

suffix => create a thread safe variable by appending thread ID to variable name

Example : Test Step Concurrency

- Achieved by putting same step sequence number in the step column of the test steps

Test Case

TestCase ID	Description	property	User	Step	Action	ActionArg_1	ActionArg_2	ActionArg_3
Login					SetContextVar	Handle		
					@OpenBrowserWithUrl.rb	\$URL	Handle	
				1	@SetTextByName.rb	%Handle%	\$USERNAME_FIELDNAME	\$MYUSERNAME
				1	@SetTextByName.rb	%Handle%	\$PASSWORD_FIELDNAME	\$MYPASSWORD
					@ClickButtonByID.BAT	%Handle%	\$LOGIN_BUTTON_ID	
					UnSetContextVar	Handle		

Molecule

Molecule ID	Description	Property	User	Step	Action	ActionArg_1	ActionArg_2	ActionArg_3
Login					@OpenBrowserWithUrl.BAT	\$input_arg2	\$input_arg1	
				1	@SetTextByName.BAT	%input_arg1%	\$YAHOO_USER_FIELDNAME	\$input_arg3
				1	@SetTextByName.BAT	%input_arg1%	\$YAHOO_PASSWD_FIELDNAME	\$input_arg4
					@ClickButtonByName.BAT	%input_arg1%	\$YAHOO_LOGIN_BUTTON_NAME	

Thread Safe Context Variables

- When multiple test cases or molecules run concurrently creating or updating a context variable with the same name, race conditions can corrupt the variable
- CHUR supports the notion of a thread specific instance of a context variable.
- Test suite designer is responsibility for creating thread specific context variables
- A **##** (double hash) appended to the name of a context variable implies a thread safe instance

Thread Safe Context variable

Comment:							
Login		GCE		SetContextVar	handle##		
				Zbrowser.Initialize	handle##	\$\$Browser	
				Zbrowser.GoToURL	%handle##%	\$URL	
				Zbrowser.SetTextByName	%handle##%	\$EmailFieldName	\$emailID
				Zbrowser.SetTextByName	%handle##%	\$PasswdFieldName	\$password

Exercise : Concurrent Execution

- Enhance yesterday's test case to concurrently search a string in Google, Bing and Yahoo.
- If you were working for Google, your test should fail if Google search is slower than other engines. How would you verify this?

Command Line Options: -include

● -Include=<location>

- This option specifies the location from where ZUG will pick up molecules and macros sheet
- Provides a dynamic way of including external molecule or macro definitions
- Test suites in the -include option must be comma separated.
e.g.

```
runzug E:\testsuites\testCMDInclude.xls -verbose  
-include=IncludeS2.xls,IncludeS1.xls
```

- The files included from command line have higher priority than files included from the config sheet.

Command Line Options: -include

- **-Include=<location>**

Provide namespaces to sheets included from command line (at run time).

Can override default namespace (file name) of a worksheet included at run time using the -include option.

Syntax for defining namespace in the -include option is

include=namespace1#filename1,namespace2#filename2

-include Exercise

Your testsuite refers to an include file called AppCfg, which is used as the namespace for the environment macros.

You want to run this testsuite on multiple environments, but you don't want to change anything in your testsuites, because they have already been debugged.

How would you go about achieving this?

-include Exercise solution

You would keep two files called QA1-AppCfg.xls and QA2-AppCfg.xls, each containing environment specific definitions of the macros.

In Zermatt, for the testsuite(s) you would configure the topology set specific command line options as follows:

- for QA1: `include=AppCfg#QA1-AppCfg.xls`
- for QA2: `include=AppCfg#QA2-AppCfg.xls`

Command Line Options - LogFileName

- **-LogFileName=<AlphaNumeric>**
- Creates log files with the specified names
<logfilename>-Result.log , <logfilename>-Error.log.

Command Line Options - Macros

• **-\$<macroname>=<value>**

-\$<macroname>={<value1>,<value2>}

- Overrides the value of a macro specified inside the spreadsheet
- If the macro name is not declared in the spreadsheet then Zug creates a new macro.
- e.g. -\$DBpwd=blah

ZUG Command Line

● **-AtomPath=<location>**

- Location from where ZUG will pick up atoms for Test Automation/Execution.
- This should be a fully qualified location.
- We can give multiple locations by ; separator.

Example: -Atompath=C:\Tests\Atoms

● **-verbose** option is used to display the execution result of the test suite in the console.

● **-TestCaseID=Test001,Test002**

- If specified Zug will execute only specific test cases as listed and will ignore the rest

ZUG Command Line

• **-Verify | -NoVerify**

- -Verify specifies Zug to execute the test case without verification.
- By default -Verify is selected meaning ZUG will run verifications for each test case unless explicitly -NoVerify is specified

• **-TestCycleID=<integer>**

- Searches in Zermatt for the specified TestCycleID.
- If it is not provided then Zug will generate a new ID and update the results under it.

ZUG Command Line

● **-TestPlanID=<integer>**

- Uploads test Suite execution results in Zermatt under a particular Test Plan.

Example: -TestPlanID=84

● **-TopologySetID=<integer>**

- Required by Zug to register results in a test cycle for the specified Topology Set.

Example: -TopologySetID=79

For More information please refer to ZUG user manual

Troubleshooting

Troubleshooting – Davos connection

• Davos connection problem

- Check the IP and port number of Davos.
- Check whether Davos is running at the correct port of the given address

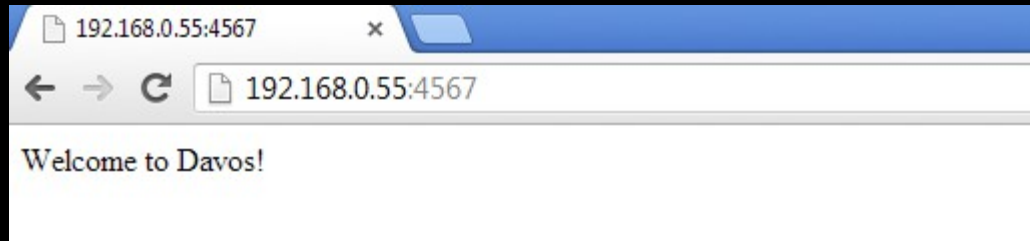
```
Current date is : 2013-6-28
Expiry date is : 2014-1-1
Zug is Valid Automature LLC. This Generic License is issued to Automature and Manimitra for usage of Zug.
Reading the TestCases Input Sheet E:\testsuites\mytestsuites\Template.xls.

SUCCESSFULLY Read the TestCases Input Sheet E:\testsuites\mytestsuites\Template.xls
Connecting to DAVOS --> URL: http://192.168.0.31:4567 User: davosuser
Controller/ConnectToDavos : Exception while Connecting to Davos Framework of host http://192.168.0.31:4567 with user d
avosuser. The Exception is DavosClient DavosAPI Error: No Response from Davos Server. Check the Url and Port.
com.automature.davos.exceptions.DavosExecutionException: DavosClient DavosAPI Error: No Response from Davos Server. Ch
eck the Url and Port.
    at com.automature.davos.engine.DavosClient.createSession(DavosClient.java:126)
    at com.automature.davos.engine.DavosClient.<init>(DavosClient.java:43)
    at com.automature.zug.reporter.DavosReporter.connect(DavosReporter.java:127)
    at com.automature.zug.engine.Controller.main(Controller.java:1510)
```

Troubleshooting – Davos connection

- **Open a web browser and enter the same URL (IP:PortNO of davos)**

-

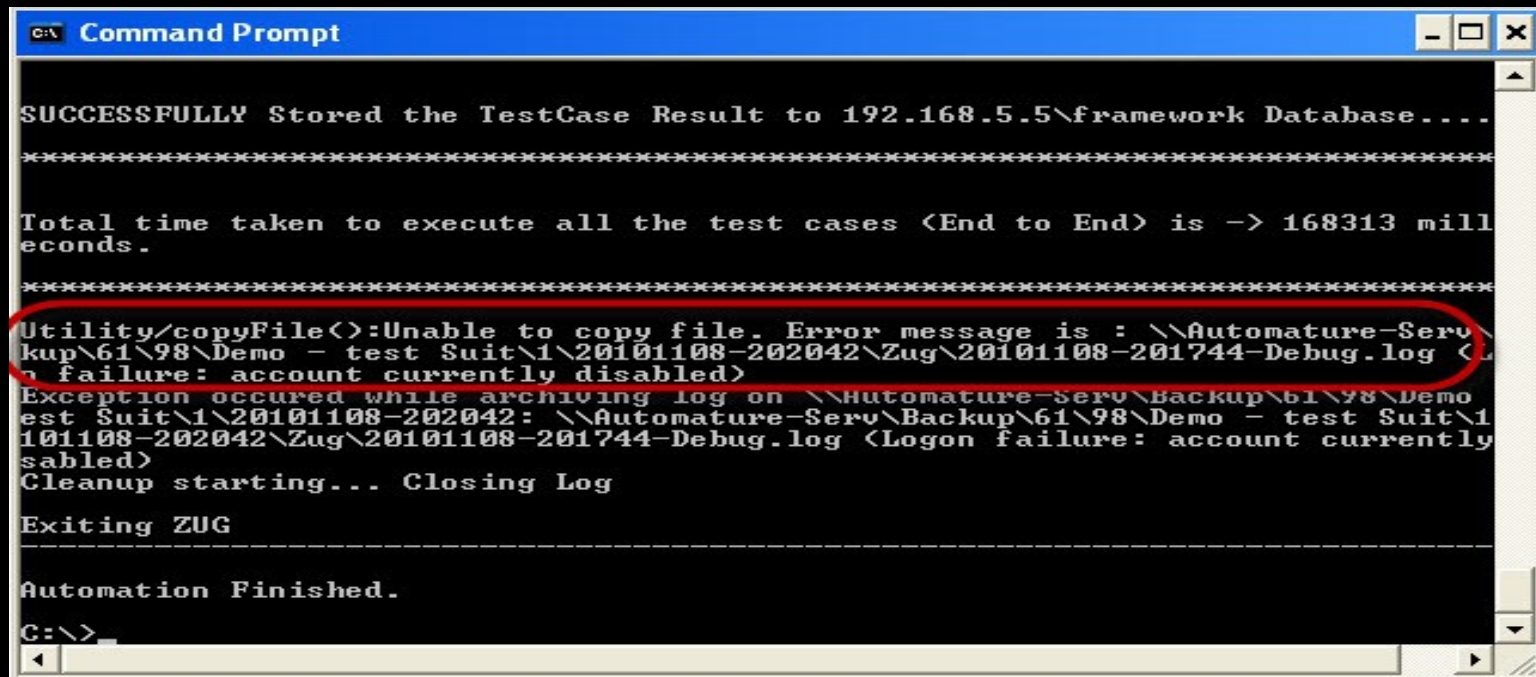


- **If Davos is running then the above response should be visible in the browser**

Troubleshooting

• Unable to Archive Log Files

- Archiving of Log files is done to location specified in Zermatt Configuration.
- Problem usually happens if the user does not have write permission for the location specified



The screenshot shows a Windows Command Prompt window titled "C:\> Command Prompt". The text inside the window is as follows:

```
SUCCESSFULLY Stored the TestCase Result to 192.168.5.5\framework Database....  
*****  
Total time taken to execute all the test cases <End to End> is -> 168313 milliseconds.  
*****  
Utility/copyFile():Unable to copy file. Error message is : \\Automature-Serv  
kup\61\98\Demo - test Suit\1\20101108-202042\Zug\20101108-201744-Debug.log <  
n failure: account currently disabled>  
Exception occurred while archiving log on \\Automature-Serv\Backup\61\98\Demo  
est Suit\1\20101108-202042: \\Automature-Serv\Backup\61\98\Demo - test Suit\1  
101108-202042\Zug\20101108-201744-Debug.log <Logon failure: account currently  
sabled>  
Cleanup starting... Closing Log  
Exiting ZUG  
-----  
Automation Finished.  
C:\>
```

The error message is highlighted with a red oval. The text "automature" is visible in the bottom right corner of the window.

Troubleshooting

● License Expired

- License can expire under the two following cases
 - You are using an expired license. Each license has a certain validity period and it expires on a particular date as mentioned.
 - You are using an improper license. The license may be incorrect or intended for other use.

```
Controller/Main : Validating Command Line Arguments
```

```
Controller/Main: Command Line Arguments Validated
```

```
Current date is : 2010-11-1
```

```
Expiry date is : 2010-11-1
```

```
Controller/Main: The License of ZUG has expired. Please renew.
```

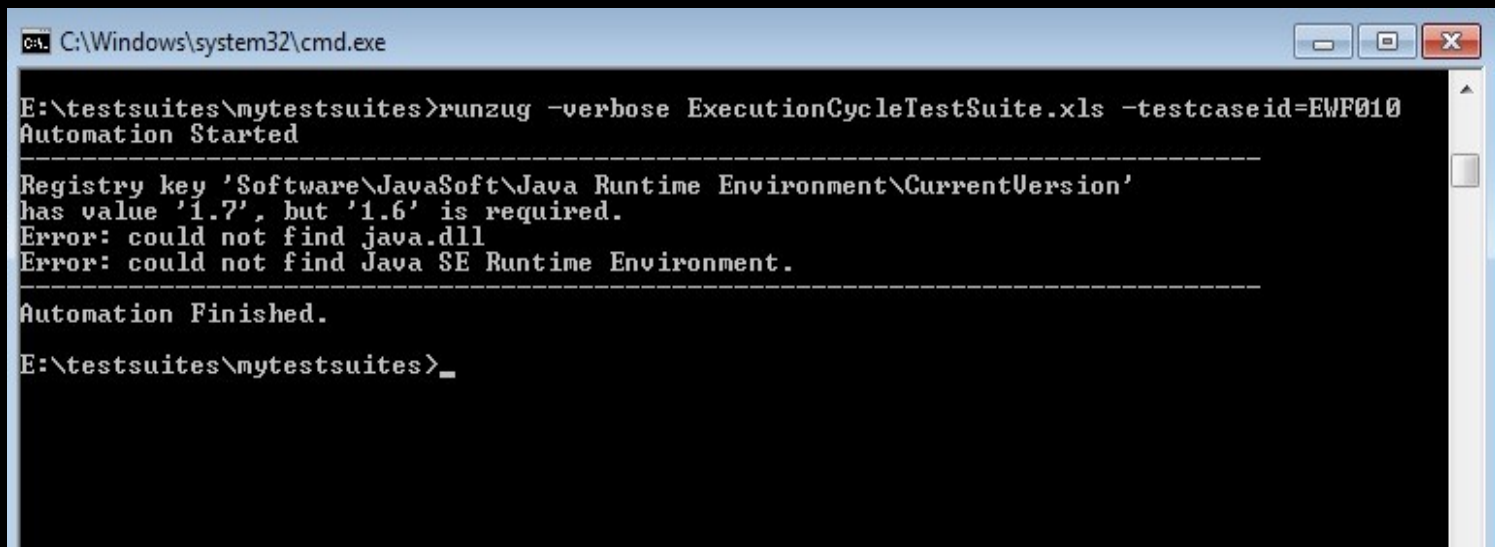
```
Visit www.automature.com
```

```
-----  
Automation Finished.
```

Troubleshooting

● JRE Dependency

- If the JRE is updated from java 1.6 to java 1.7 after installing ZUG then ZUG will stop working and it will give an error message similar to the message given below

A screenshot of a Windows command prompt window. The title bar reads 'C:\Windows\system32\cmd.exe'. The command prompt shows the following text:

```
E:\testsuites\mytestsuites>runzug -verbose ExecutionCycleTestSuite.xls -testcaseid=EWP010
Automation Started

-----
Registry key 'Software\JavaSoft\Java Runtime Environment\CurrentVersion'
has value '1.7', but '1.6' is required.
Error: could not find java.dll
Error: could not find Java SE Runtime Environment.
-----

Automation Finished.

E:\testsuites\mytestsuites>_
```

- **Solution:** Reinstalling ZUG will fix the problem

Troubleshooting

• Java Heap Error

- Make Sure that you are using the **client** JRE.
- Type `java -version`

cmd Select C:\Windows\system32\cmd.exe

```
C:\Users\skhan>"C:\Program Files\Java\jre7\bin\java" -version
java version "1.7.0_09"
Java(TM) SE Runtime Environment (build 1.7.0_09-b05)
Java HotSpot(TM) 64-Bit Server VM (build 23.5-b02, mixed mode)
```

- If it shows Server VM please install a the client VM (Java).

```
C:\Users\skhan>java -version
java version "1.7.0_21"
Java(TM) SE Runtime Environment (build 1.7.0_21-b11)
Java HotSpot(TM) Client VM (build 23.21-b01, mixed mode, sharing)
```


Multi-environment testing

Multi-environment testing

- We can select dataset for target specific testing.
- Zug allows us to have multiple value columns in the macro sheet of a test suite
- Each value column of the macro sheet is a data set
- When running the test suite user can choose a particular value column of the macro sheet of a test suite whose value will be used while running the test suite.

Multi-environment testing

- If any cell of the selected column is empty then it takes the value of the default column cell (1st value column)

1	Macro Name	Value		Value	Value
2	\$LOGIN	/twiki/bin/login/			
3	\$LOG_OUT_TEXT	Log Out			
4	\$WEB	ZERMATT			
5	\$DOMAIN	192.168.0.31		192.168.0.55	192.168.0.2
6	\$BROWSER	htmlunit		chrome	ie
7	\$LOGIN_NAME_VALUE	admin			PatAdmin
8	\$PASSWORD_VALUE	automature		@dmin	
9					
10	\$PRODUCT_FIELD_VALUE	[MARS:first test:nothing:none:none]		[JUPITER:first test:nothing:none:none]	[EARTH:first test:nothing:none:none]

Multi-environment testing

- Select a particular column at runtime using the –macrocolumn command line option.

Syntax for -macrocolumn is

-macrocolumn=fileidentifier:column_no, fileidentifier:column_no

- File identifier is the file name or the optional name space provided in the -include option

EX:

E:\testsuites\testMacroColumnNS.xls

-include=IncludeS2.xls,inc#IncludeS1.xls

-macrocolumn=inc:3,testMacroColumnNS.xls:2,IncludeS3.xls:3

Maintainability & design

Maintainability & design - goals

- Improve **portability**

 - Executable on multiple test machines

 - Allow testing of multiple target environments

- Improve **organization**

 - Group common elements together, and separate different elements (e.g. Molecules that do related things)

- Improve **readability**

 - Consistent & meaningful naming conventions

 - Use of local versus global context variables

 - Use of molecules to encapsulate logic

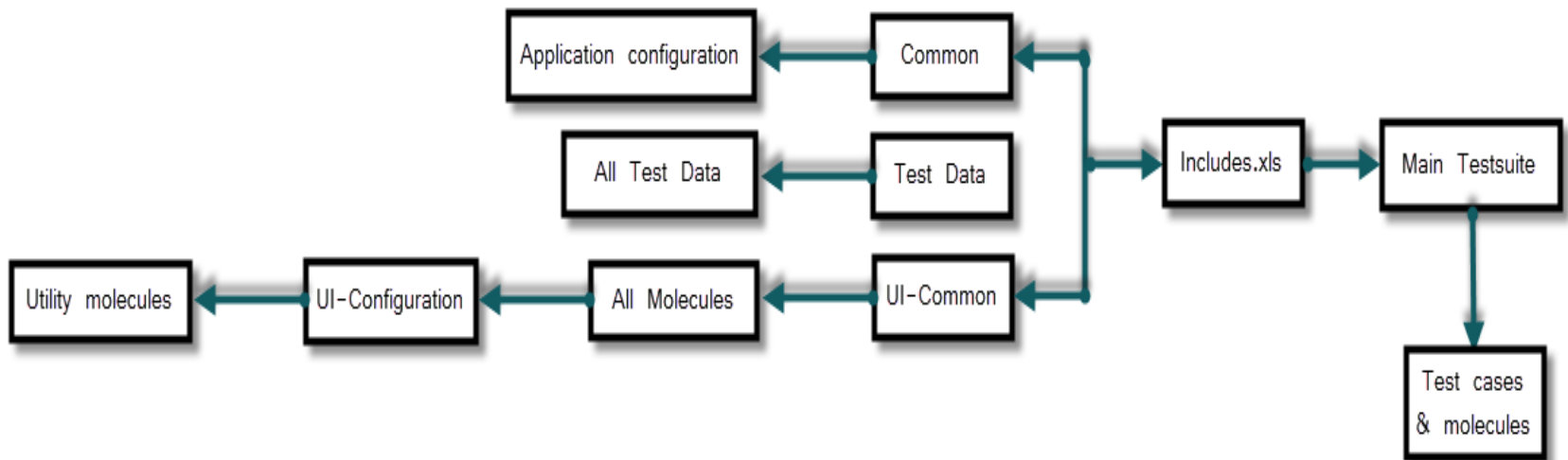
- Improve **performance**

 - Pay attention to thread safety

Maintainability & design

- The test suite should be designed in such a manner that it is easily maintainable.

Test Suit Design & Maintainability Workflow



A Case Study: UI-BPO

- Common folder contains Application & Topology specific configurations which are common to all other test suites.
- TestData folder contains all the data which are meant for creating fake content. e.g. Fake customer profiles, testfiles
- UI-Common folder contains all other molecule sheets, UI-Test configuration data and reusable molecules (Utility molecules).
- Includes.xls includes all the external worksheets required by the actual testsuite.
- Main Testsuite includes Includes.xls, Testcases & molecules.