DRY

OUTRIGHT FRAMEWORK

Outline

Notes on UI Automation

Motivation

Structure

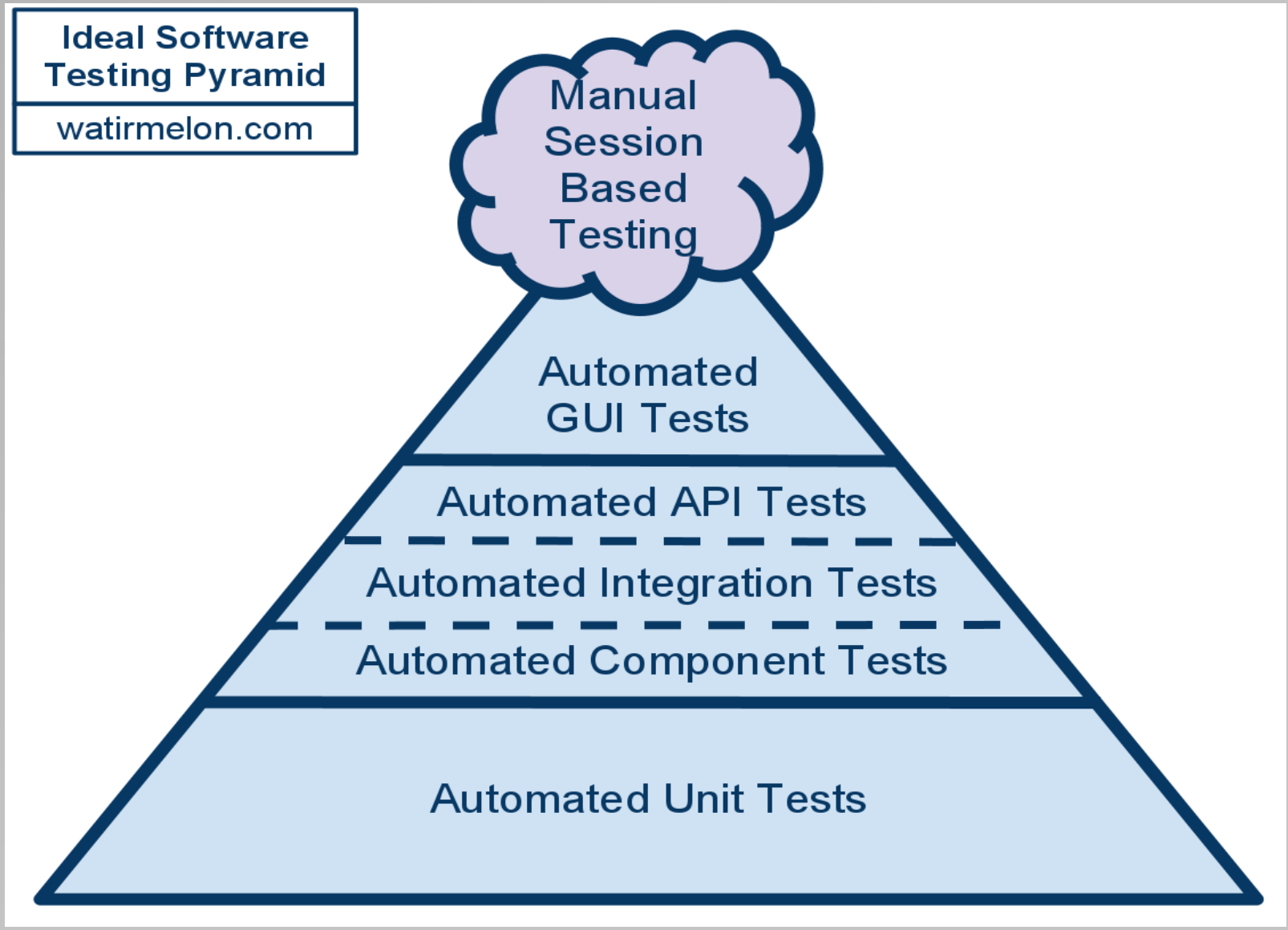
Function

Tool Box

Notes on UI Automation

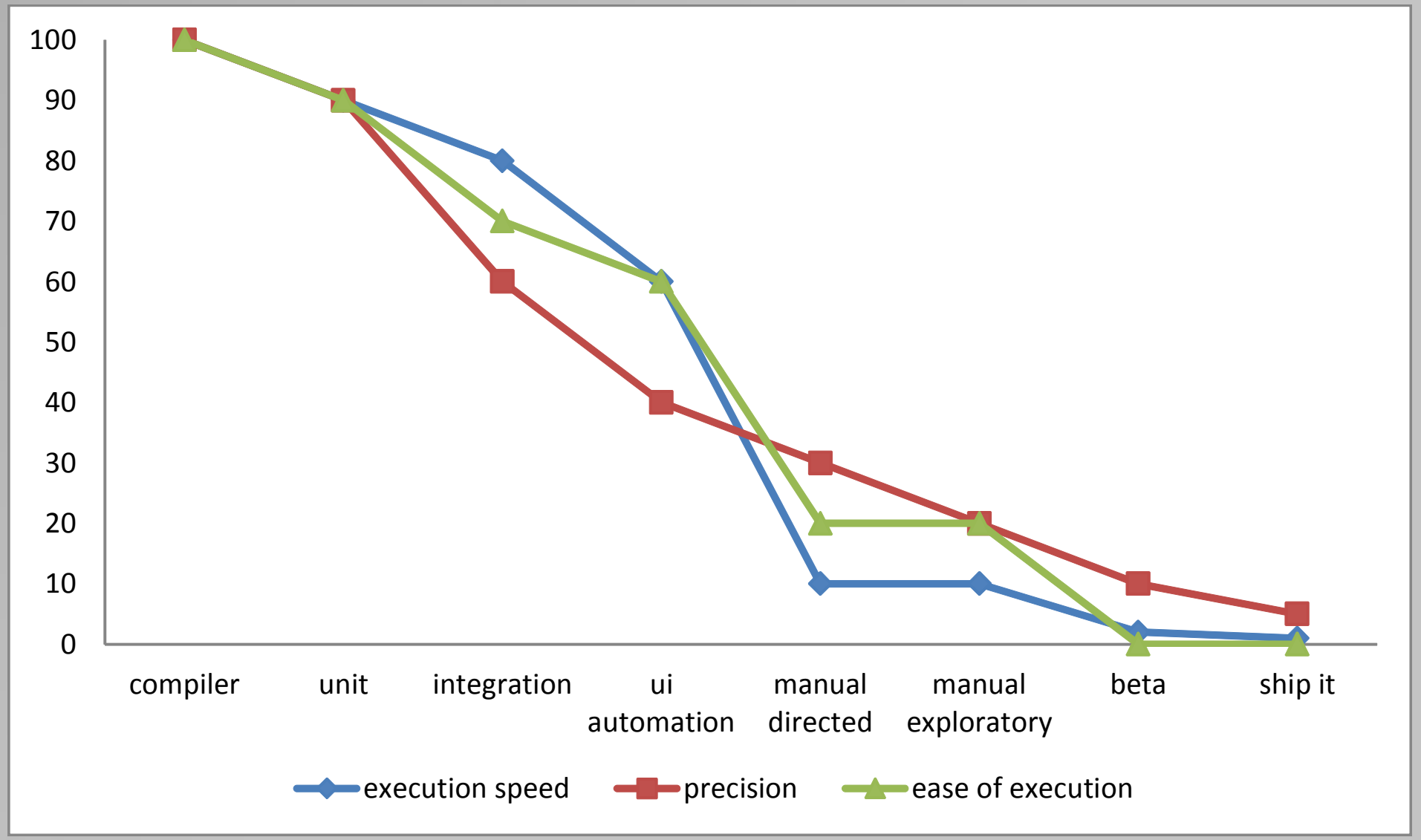
CHARACTERISTICS

Functional Test Pyramid

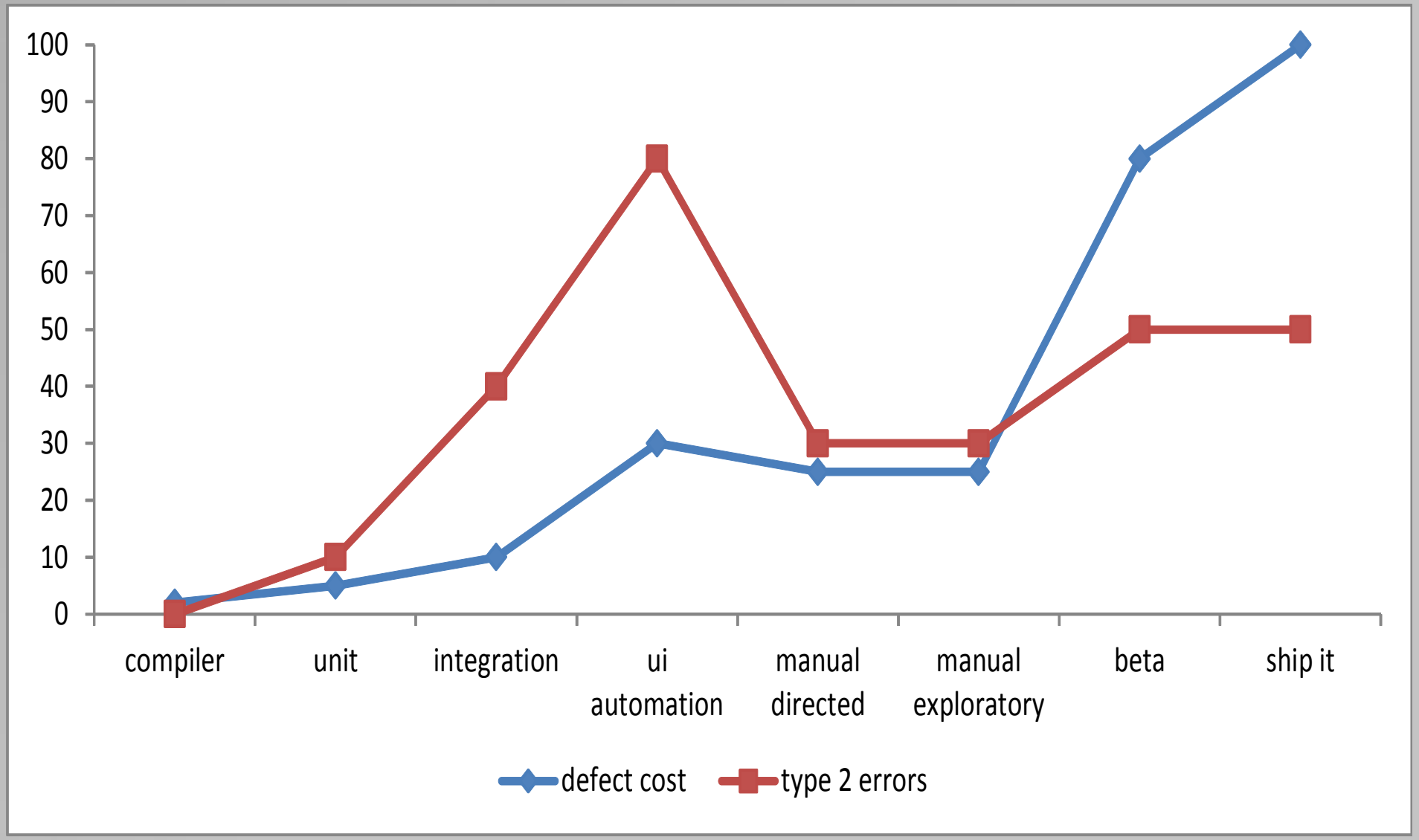


Test Automation Basics – Levels, Pyramids & Quadrants

CHARACTERISTICS

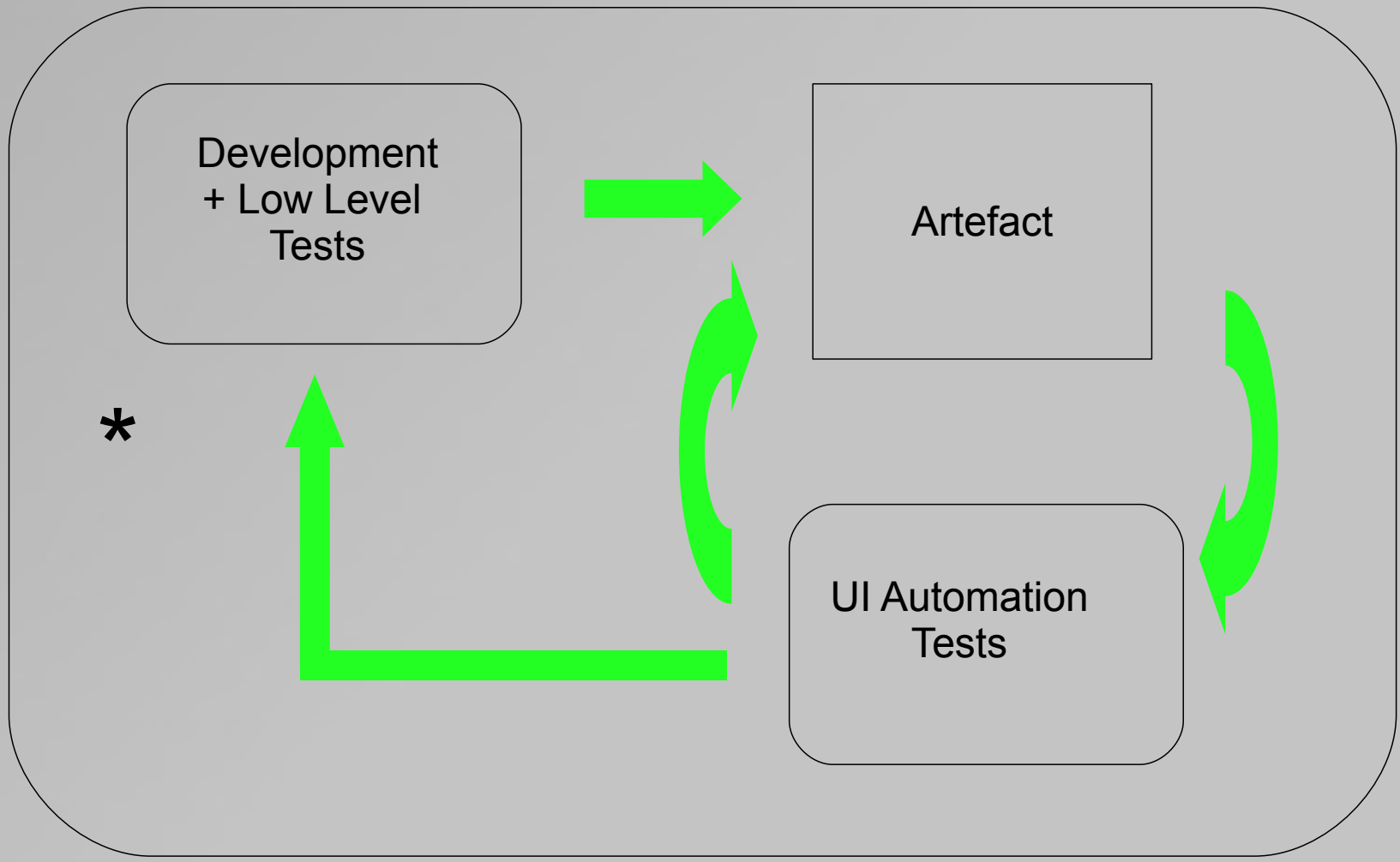


CHARACTERISTICS



Separation of Concerns

CHARACTERISTICS



* Conditions Apply

UI Automation Key Points

CHARACTERISTICS

- Is more difficult to maintain and run than low level tests
- Is slower to run than low level tests
- Is magnitudes faster than manual testing
- Is more prone to Type 2 errors than low level tests
- Is more likely to surface defects in the AUT that are incidental to the feature being targeted by the test
- If owned by QA proved a valuable control outside development

Poorly Implemented UI Automation

CHARACTERISTICS

- Is extremely difficult / impossible to maintain
- Is excessively prone to Type 2 errors
- Is usually abandoned soon after implementation

Motivation

DRY

Automation === Software Development

ANTIPATTERNS

- The magic tool
- The opaque facade
- Simply record and playback

- Self taught QAs
- The seconded Dev => Dev dependency
- The vanishing consultant
- Training then nothing

- Starting from scratch
- Automate everything
- Integrate everything

- DevOps not considered
- Who owns this?

Consequences

ANTIPATTERNS

=> NOT DRY && NOT UNDERSTOOD

=> UNMAINTAINABLE

=> NOT MAINTAINED

OUTRIGHT FRAMEWORK

To Become Visible Service Outright Trickery

The only UI automation frameworks that have a reasonable chance of success are code (aka script) based.

Although UI automation development need not be executed with the same rigour as the development of production code, neglecting software development fundamentals when building an automation suite will invariably lead to failure.

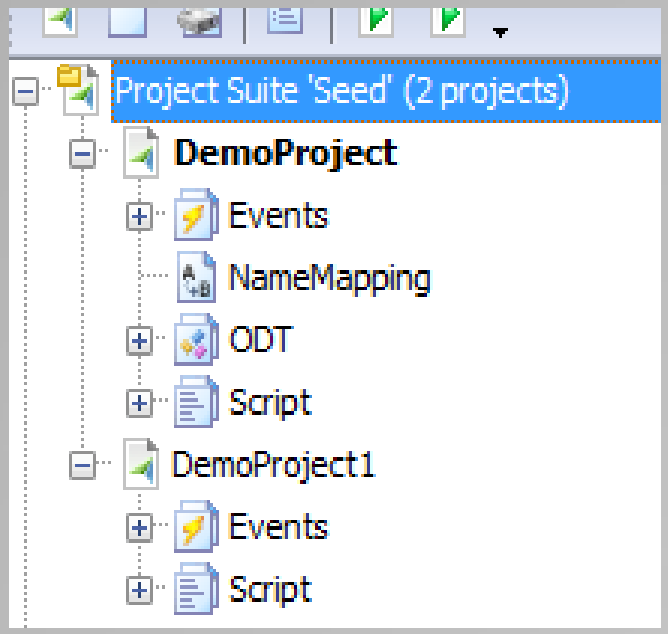
The OutRight framework is a set of script based utilities and conventions that implement generic automation capabilities and incorporate “better” software development practices. These utilities and conventions enable the development of automation suites that are **small, understandable, testable,** and consequently, **maintainable**.

The OutRight framework is built on top of TestComplete which is a superb automation tool with excellent documentation. As an out of the box solution to many technical automation issues is provided, implementation can focus more heavily on training and mentoring. Such an approach greatly improves the chance of success.

H_0: UI Automation is Snake Oil

STRUCTURE

PROJECT STRUCTURE



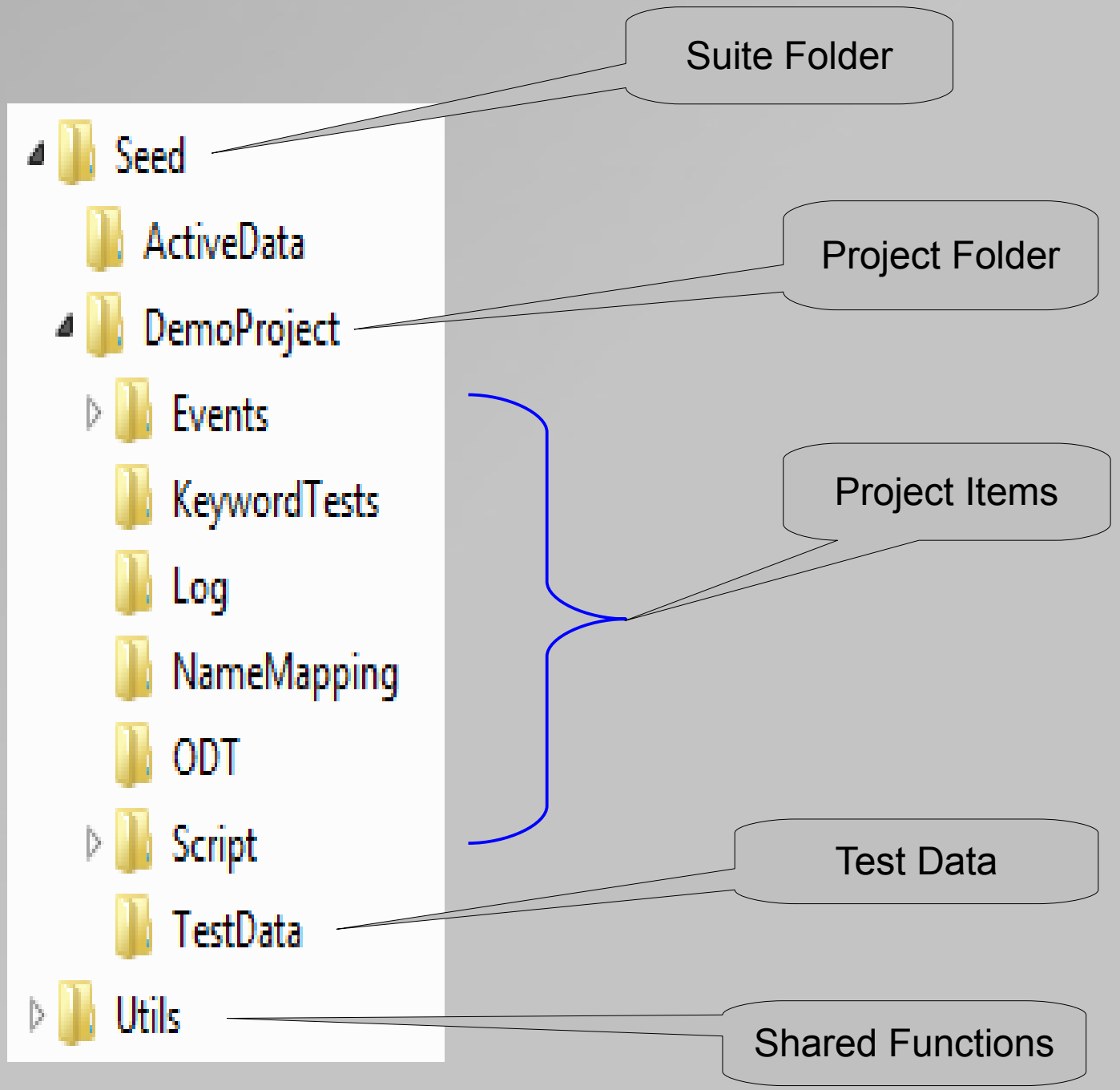
Project Suite

- Corresponds to an application under test
- May also be used to isolate a single project (for IDE performance)

Project

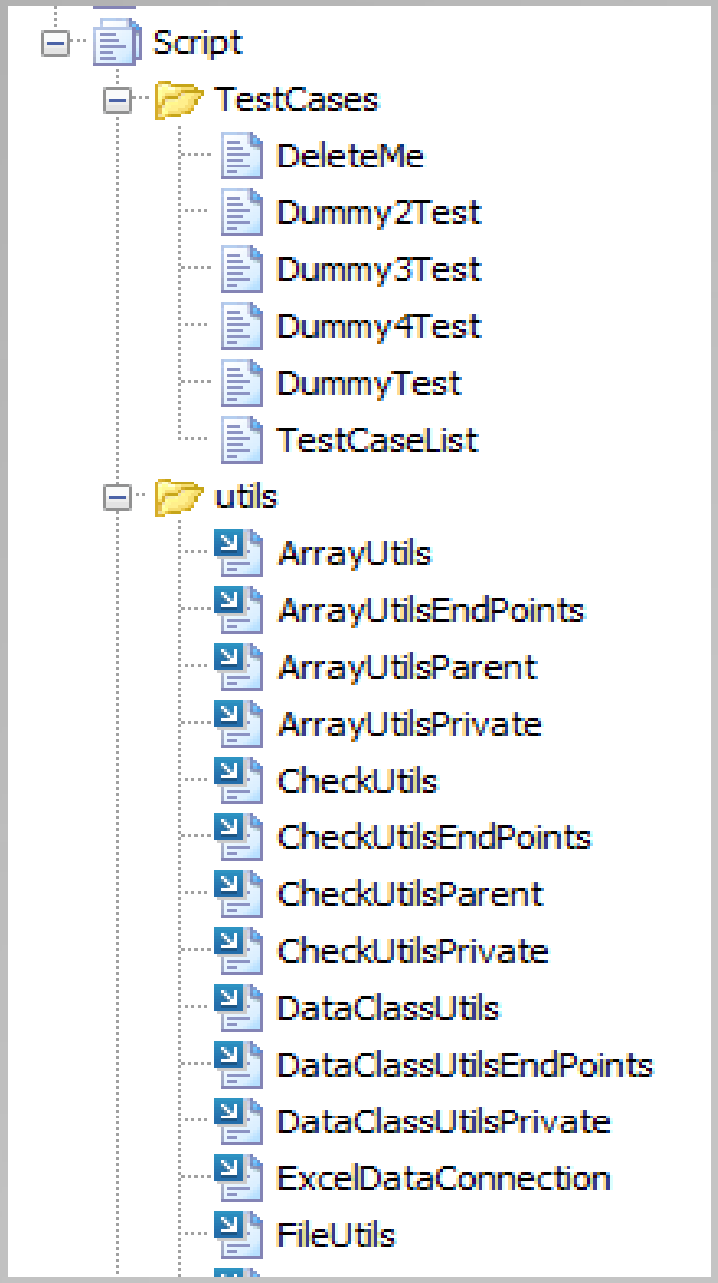
- Can be an arbitrary collection of tests or correspond to a functional area
- Having multiple projects can improve IDE performance on very large test suites
- Contains all the test items such as scripts and name mappings
- Project items can be shared between projects

FILE STRUCTURE

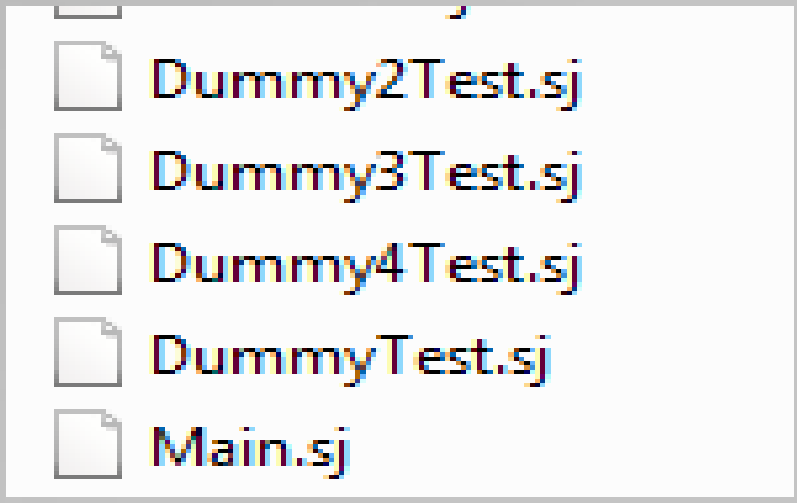


SCRIPT UNITS

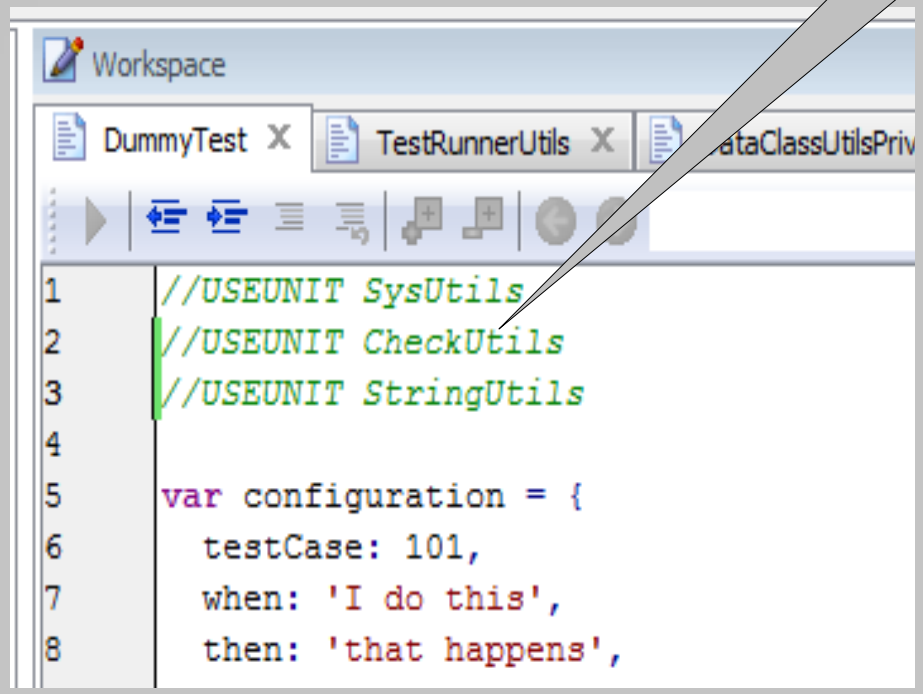
TestComplete - Project View



Files

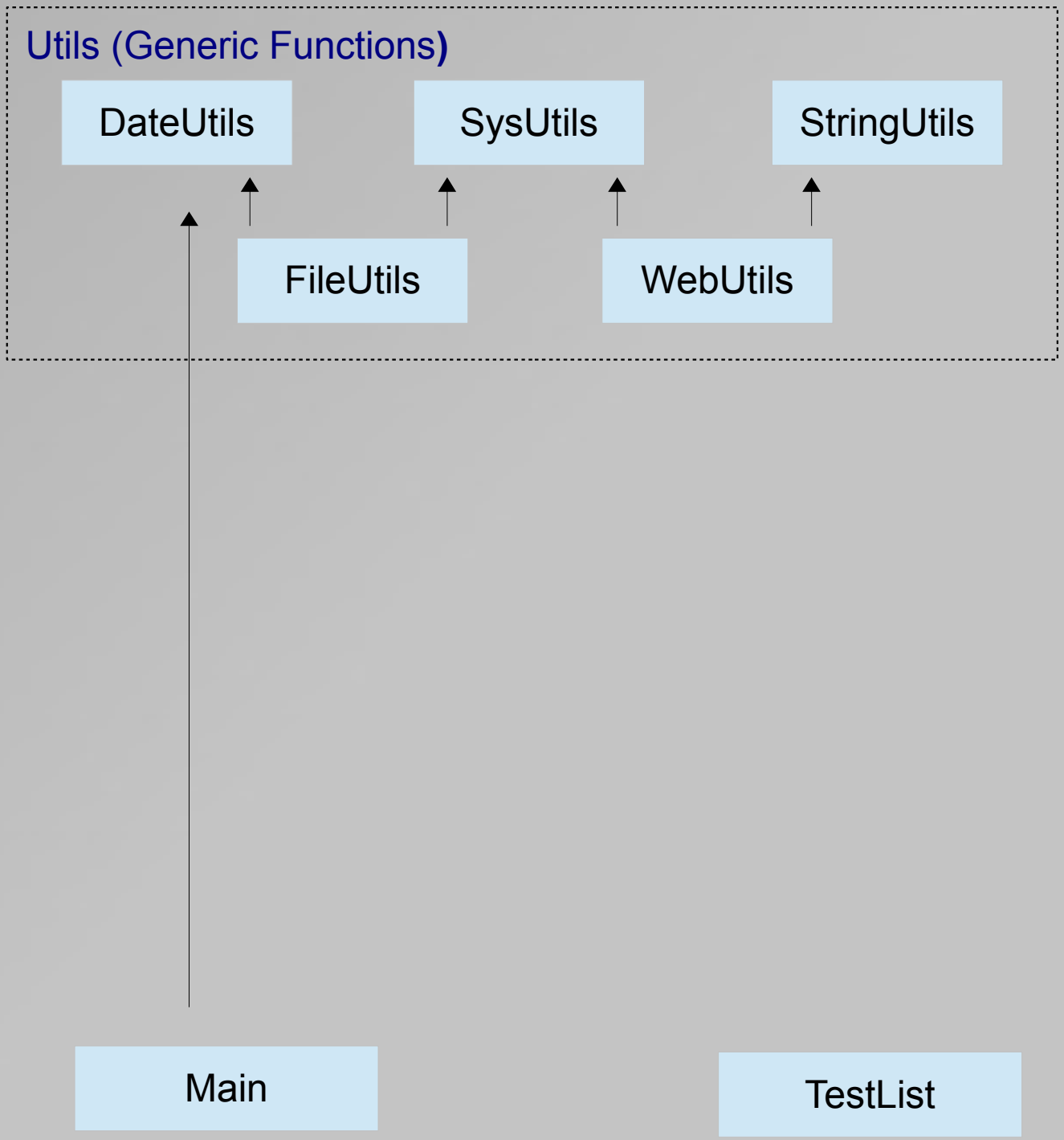


TestComplete – Script Editor

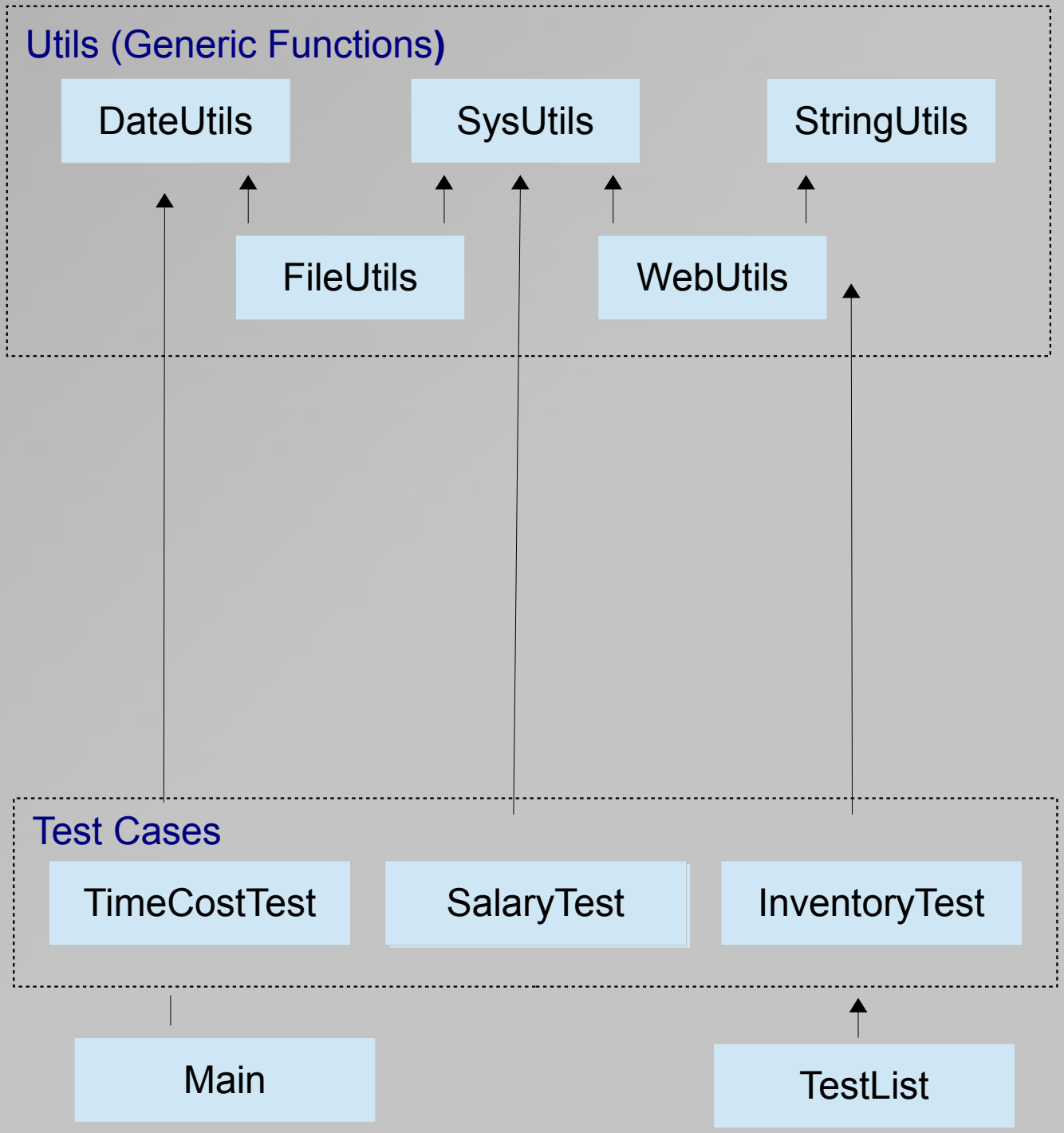


Uses Clause

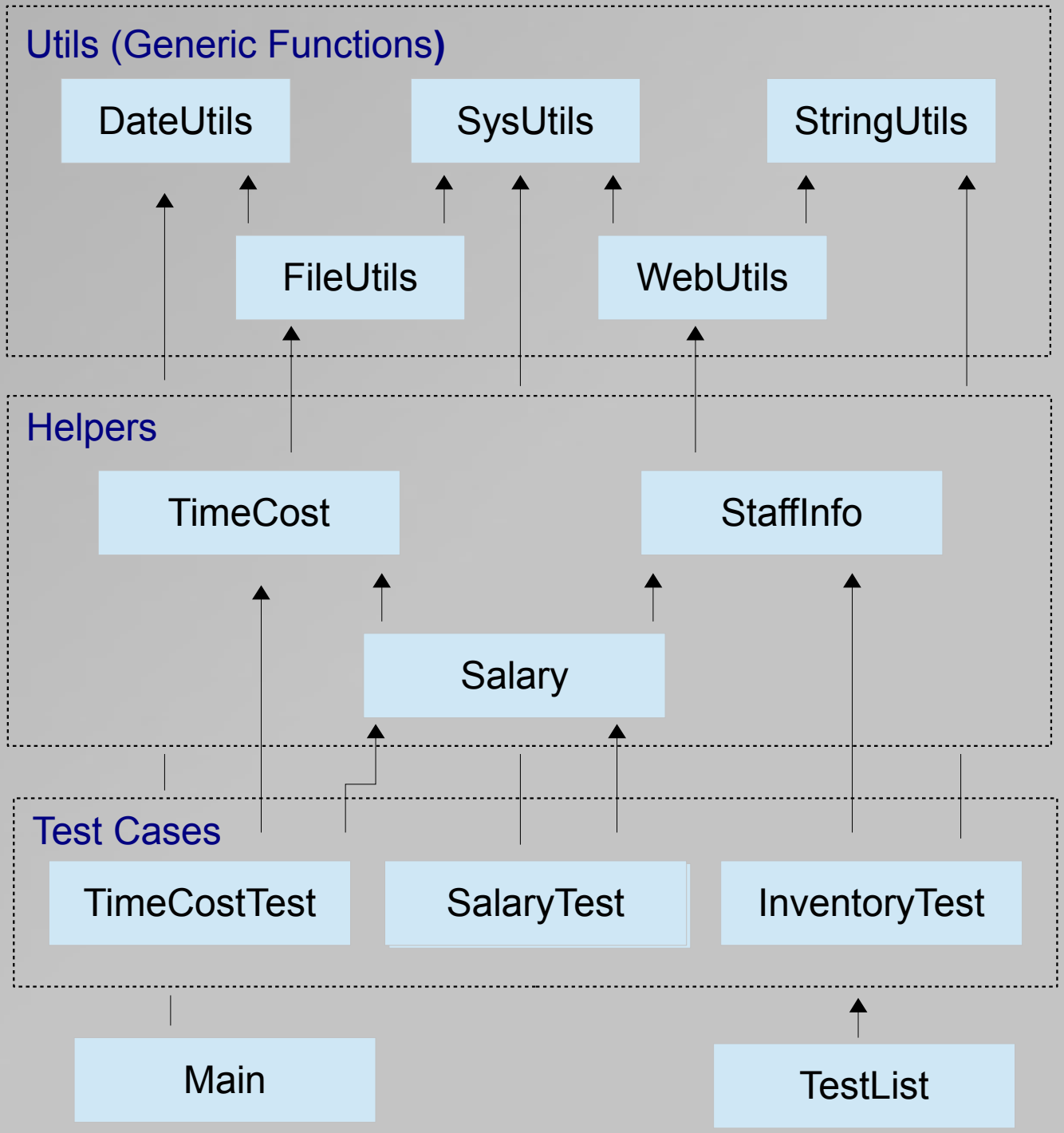
UNIT STRUCTURE



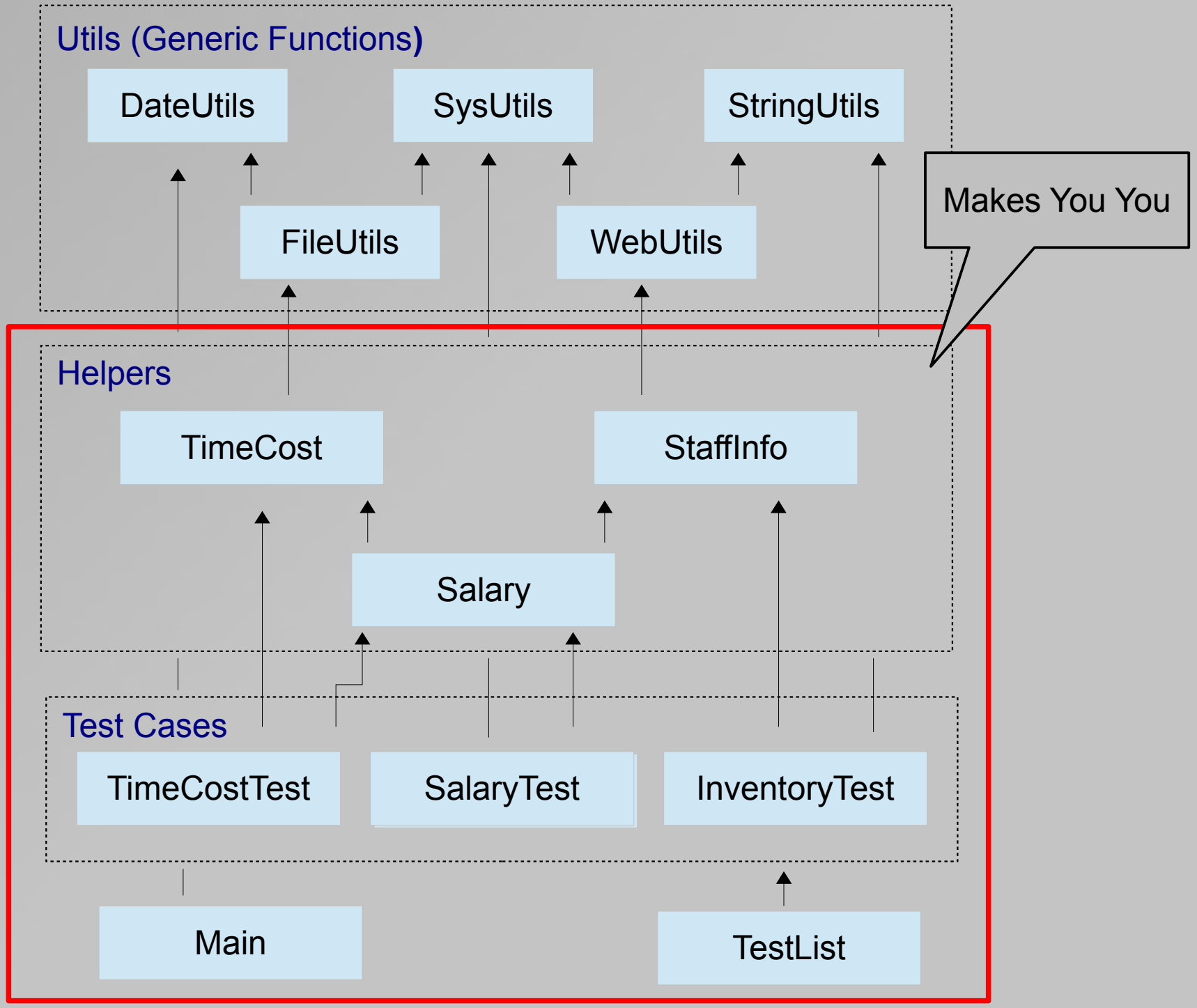
UNIT STRUCTURE



UNIT STRUCTURE



UNIT STRUCTURE



UNIT STRUCTURE

Utils (Generic Functions)

- Are the basis of the framework
- Provide generic cross-project / cross-application functions
- Can only use other utils
- Named <Functional Area><Utils> e.g. StringUtils
- Are often wrappers around TestComplete functions

Helpers

- Are normally project / application specific
- Contains code shared by two or more test cases
- Are built up as test cases are refactored
- Can use utils or other helpers
- Named <Functional Area> e.g. Salary

Test Cases

- Application project specific
- Can use utils or helpers
- Cannot be used by any other unit than TestList
- Named <TestSummary>+Test e.g. SalaryCalculationTest

Two Special Units

UNIT STRUCTURE

Main – Defines How Your Project Behaves Differently to Other Projects

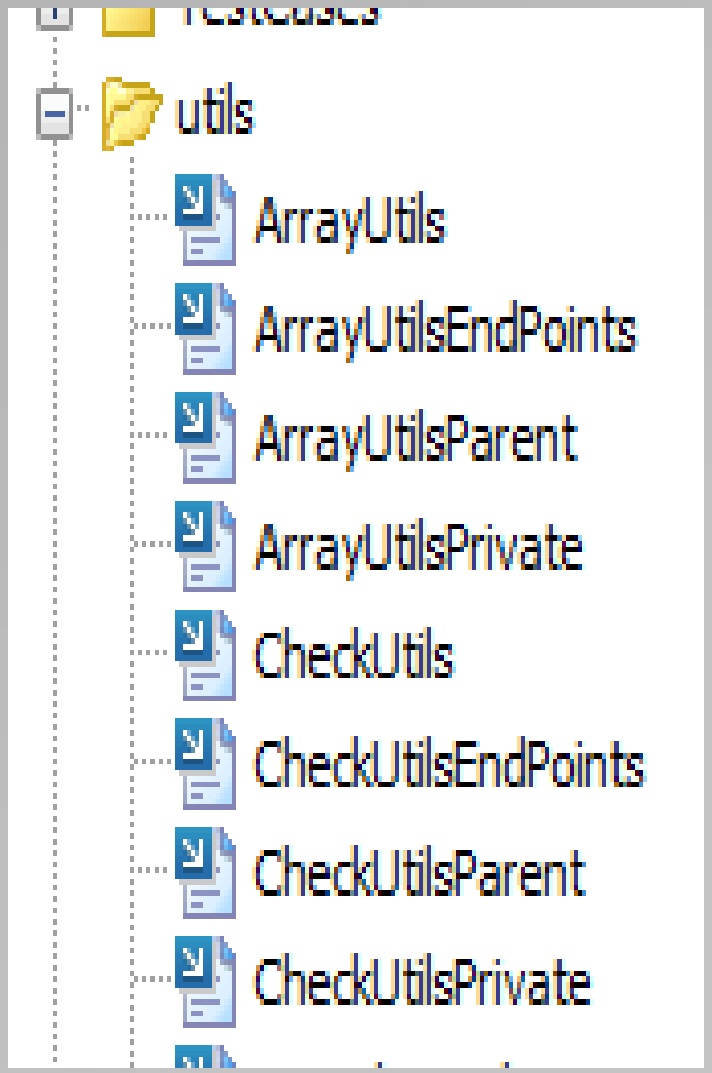
- Test Run Functions
- Test Configurations
- Links up test run controlling functions (i.e. Composition Root)
  - preAcceptMethod
  - restartMethod
  - simpleLogProcessingMethod

Test List – Partially Automated List of All Tests that Can Be Run

- Contains a list of all test units in the project
- Contains restart tokens
- Is validated and updates itself and fails if the list is not valid at the start of the test

Auxiliary Units

UNIT STRUCTURE



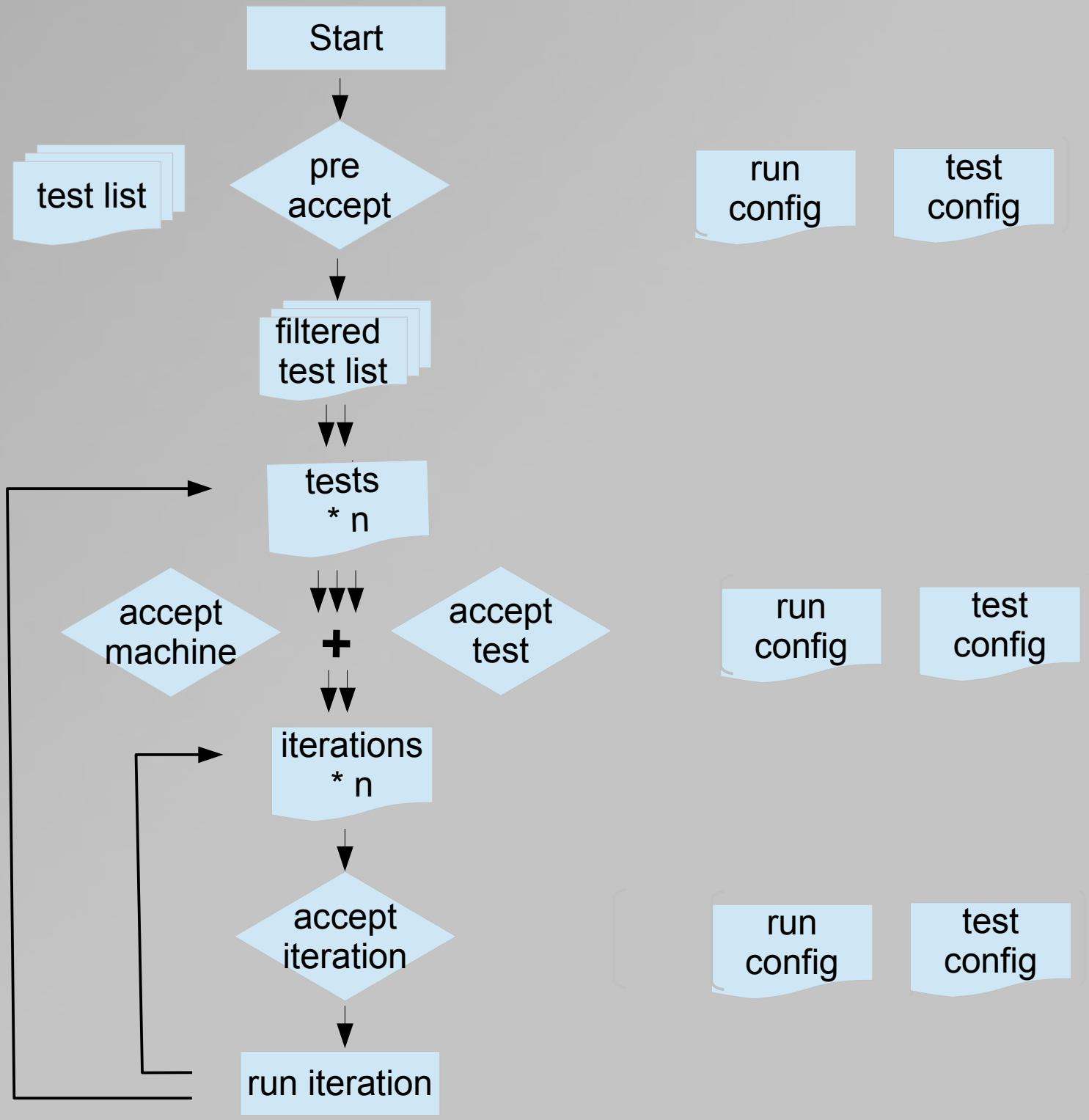
* EndPoints
- Occur with Utils and Helpers
- Contain UnitTests and Endpoints
- Reduce clutter in code completion
- Not necessary for TestCases / Priv Units

* Private
- Occur with Utils and Helpers
- Reduce clutter in code completion
- Not necessary for TestCases

* Parent / GrandParent
- Use to resolve recursive references in Utils and Helpers

FUNCTION

TEST RUN


Start
test list
pre accept
run config
test config
filtered test list
tests * n
accept machine
+
accept test
run config
test config
iterations * n
accept iteration
run config
test config
run iteration

TEST CASE FUNCTIONS

```
//USEUNIT TestRunnerHelpers

var configuration = {
  testCase: 1,
  when: 'an action is taken',
  then: 'an equal and opposite reaction occurs',
  dataClass: 'DataClassGenTest',
  country: 'All',
  owner: 'Li',
  enabled: true
  };

function testCase(runConfig, testConfig, testCaseID, iteration){
  // test case goes here
}

function testCaseEndPoint(){
  runTestCaseEndPoint(configuration.testCase);
}

function config(){return configuration}

function acceptTest(runConfig, testConfig){
  return true;
}

function acceptClientMachine(runConfig, testConfig){
  return true;
}

function acceptIteration(runConfig, testConfig, testCaseID, iteration){
  return iteration < 2;
}
```

TOOLBOX

TOOL SOURCES

- TestComplete
- TestComplete script extensions
- Outright Framework functions
- External JS libraries
- Native Javascript

DEMO

Favourite Features

TESTCOMPLETE

Help
- Language filter

Editor
- Code Navigation - unit or function
- Find / Replace
- Bookmarks
- Script search
- Code templates
- <View><Desktop><Restore Default Docking>

Script Development
- Object Spy
- Record / Playback

Debugging
- The log file
- Debugger / debugger windows

Favourite Features

TESTCOMPLETE

Help
- Language filter

Editor
- Code Navigation - unit or function
- Find / Replace
- Bookmarks
- Script search
- Code templates
- <View><Desktop><Restore Default Docking>

Script Development
- Object Spy
- Record / Playback

Debugging
- The log file
- Debugger / debugger windows

aq* - TestComplete (initially developed by Automated QA)

- aqDateTime – date time calculations
- aqEnvironment – info about operating system
- aqFileSystem – working with files
- aqFile – read write files complements
- aqConvert – data conversion
- aqString – string functions
- aqObject – reflection

Other Useful Objects

- Log – write to log file (also in utils)
- Indicator – show information in indicator
- Runner – run methods (advanced)

OUTRIGTH FRAMEWORK

Favourite Features

Generic Features

- TestCaseList & restarts
- Even easier data driven testing
- End points / unit tests
- testCaseEndPoint
- OTB code templates
- API Docs
- simpleLogFileProcessing

Methods

- preAccept
- def
- seek, seekh, seekInPage, seekInPageh
- expectDefect / endDefect
- stringToFile / arrayToFile

EXTERNAL LIBRARIES

Underscore.js

JSON2

NATIVE JAVASCRIPT

w3schools

Javascript the Good Parts