Software Testing

Automation Test

Nguyễn Đạt Thông

www.ndthong.info

HCMUS-VNU

Course Introduction
Content and Approach
Evaluation and Examination

INTRODUCTION

Course Information

- Name: Automation Test
- Duration
  - 30 periods - lecturing
  - 30 periods - practicing
- Instructor: Nguyễn Đạt Thông
  - Email: ndthong@math.hcmus.edu.vn

Content

- Automation Test
  - Automation Test
  - Automation Test with Keyword
- Automation Testing Tools
  - Selenium
  - TestArchitect™
  - TestComplete

References

- Internet Search
  - Automation test
  - Automation test keyword
  - Action based testing

- Provider's Sites
  - Selenium HQ – http://docs.seleniumhq.org
  - TestArchitect™ – http://testarchitect.com
  - TestComplete – http://smartbear.com

Approach

- Lecture (10 periods)
  - Automation Test
- Group: Seminar (20 periods)
  - Automation Testing Tools
- Group: Practice (30 periods)
  - Automation Testing Tools

Evaluation

- Regular
  - Seminar: 30%
  - Practice: 30%
  - Examination: 40%

- Extra
  - Online examination: 10%

Practice

- Self-study, raise questions

- Submission on one of predefined projects
  - Web test project using a Selenium framework
  - Application test project using TestArchitect™
  - Mobile test project using TestComplete

- Submission requirements
  - Team (2-3)
  - Test cases (40), Test suites (5)

Examination

- Content
  - Automation Test
  - Tools: Basic Usage

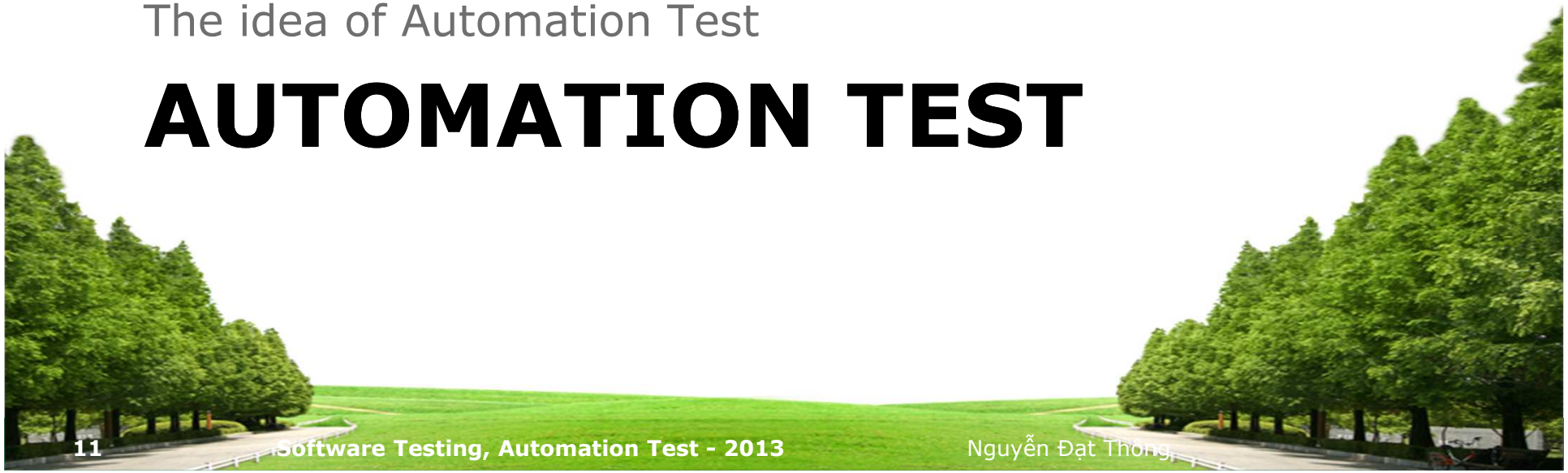
- Multiple Choices
  - 30 questions
  - 75 minutes
  - Hardcopy documents allowed

Question

Raise your hand, please.

Automation Test
Automation Test with Keyword
The idea of Automation Test

AUTOMATION TEST



Content

- Automation Test
  - Overview
  - History
  - Challenges
- Automation Test with Keywords
  - Keywords
  - Action based Testing

Automation Test

- What is Automation Test?
- Benefits and pitfalls of Automation Test
- Applying Automation Test
- Automation Test Production Process
- History of Automation Test
- Challenges of Automation Test

Objectives

- Present an overview of Automation Test
- Understand the pros and cons of automation test
- Understand the strategies and tactics in automation test

What is Automation Test?

"Using computer time to execute of tests."

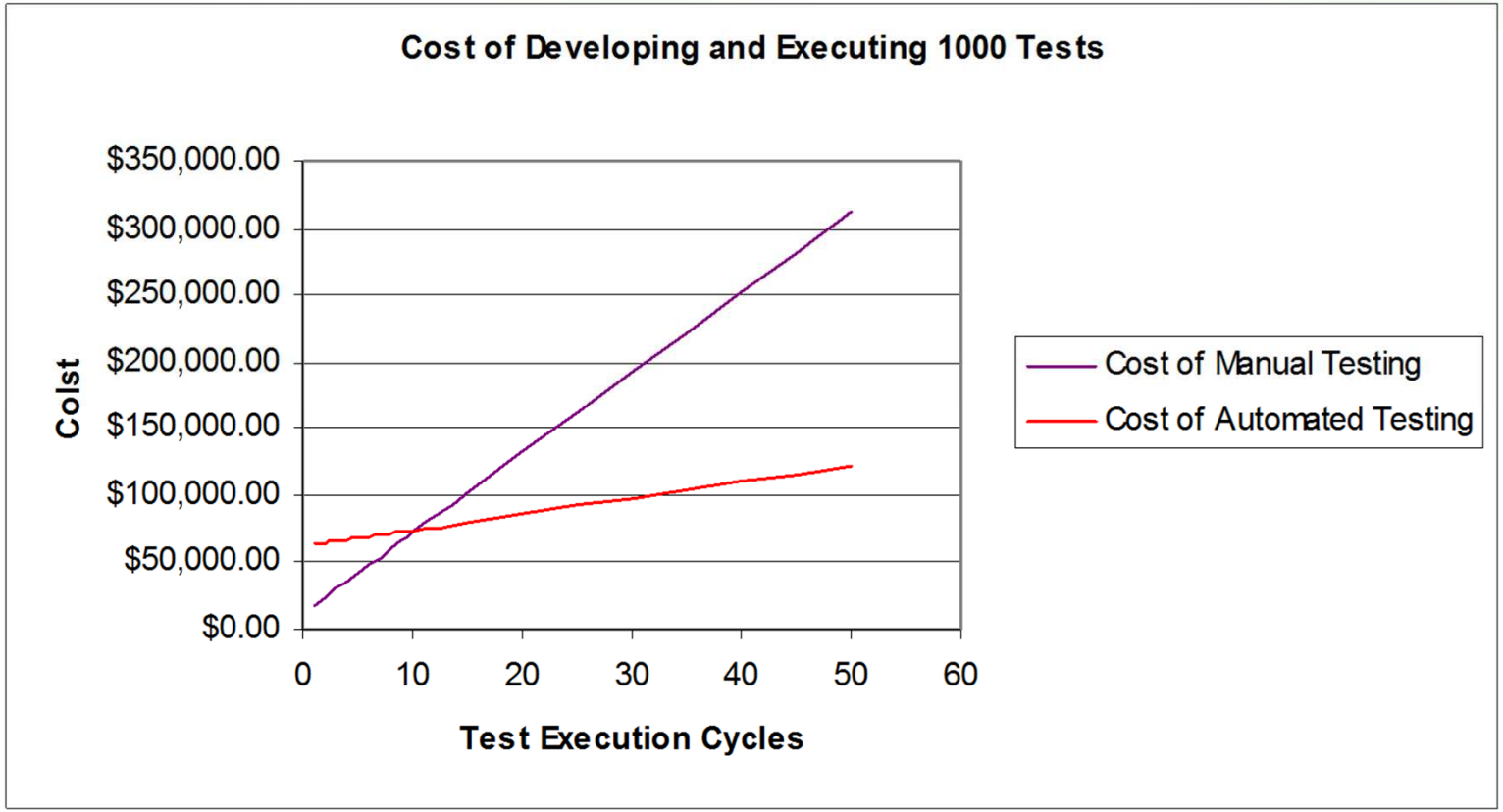
- Commonly, test automation involves automating existing and sometimes, new tests that otherwise, will be executed by a human tester

Why automate testing?

- People might say,
  "Automation will help us release a higher quality product on time."
  - What does that mean?

- What's the goal of instituting an Automation program?
  - What problem are we trying to solve?

Because of the cost?



Why?

- Repetitive tests that run for multiple builds.
- Tests that tend to cause human error.
- Tests that require multiple data sets.
- Frequently used functionality that introduces high risk conditions.

And why?

- Tests that are impossible to perform manually.
- Tests that run on several different hardware or software platforms and configurations.
- Tests that take a lot of effort and time when manual testing.

Manual vs. Automated

Manual Testing

- Test execution time ↑
- Regression test coverage ↓
- Responsive and flexible
- Human variation
- No maintenance needed
- Immediate dividend paid
- Not cost effective for high volume regression

Automation Testing

- Test execution time ↓
- Regression test coverage ↑
- Relatively inflexible
- Machine consistency
- High maintenance needed
- Long-term dividend
- Not cost effective for low volume regression

What should be automated?

"Anything that helps the testing be more efficient and adds value to the testing effort"

Tests should be automated?

- Code level tests?

  unit test, code verification, code coverage…

- API tests?

  libraries, web services…

- Performance tests?

  run-time performance comparisons

- Load/stress tests?

  putting demand on a system or device and measuring its response

- **Functional tests?** *… continued*

Tests can’t be automated?

- Usability testing

  "How easy is the application to use?“

- One-time testing

- “ASAP” testing

  “We need to test NOW!”

- Ad hoc/random testing

  Based on intuition and knowledge of application

- Tests without predictable results...

Automation is always good?

- Uncertainty and lack of control.
- Poor scalability and maintainability.
  - Too sensitive to changes
  - Poor reusability
- Low test automation coverage
  - Too much programming

… and other pitfalls

- Poor methods and disappointing quality of tests
  - Focus of tools instead production volume
  - Focus of making the tests run instead of quality of the tests

- Technology vs. people issues
  - Gaps in platform technology support
  - Gaps in human resource equipped with test automation skills

When we need automation?

- Customer requests
- Application should be stable
- Tests that need to be run for every build of the application (sanity check, regression test)
- Tests that use multiple data values for the same actions (data driven tests)

… and some criteria

- Tests that require detailed information from application internals (memory, CPU)
- Tests that require considerable amount of time to perform manually
- Stress, load or performance testing

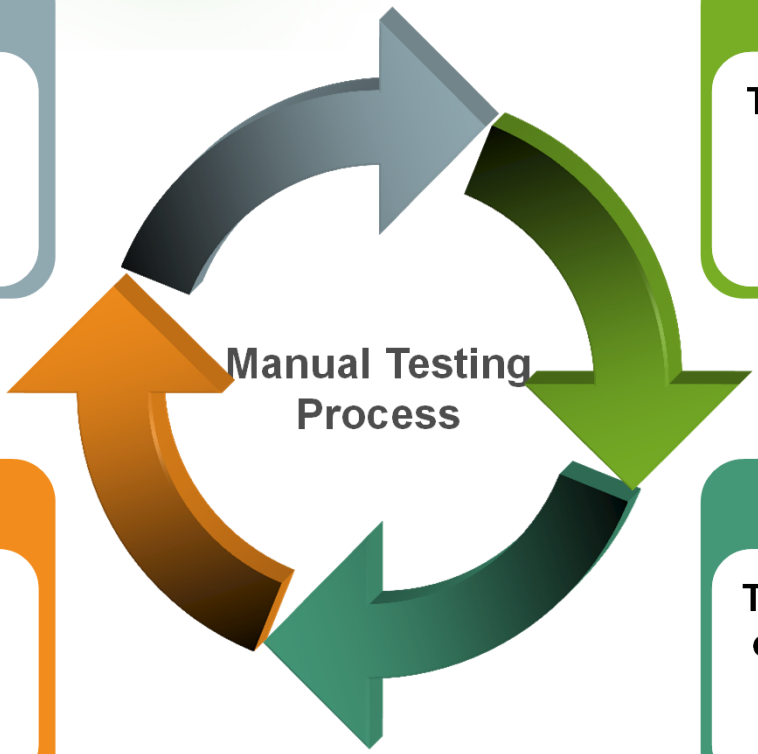
How we test manually?

Input

Test Engineers make input actions to the AUT

Synchronize

Test Engineers wait until the AUT finishes processing the inputs



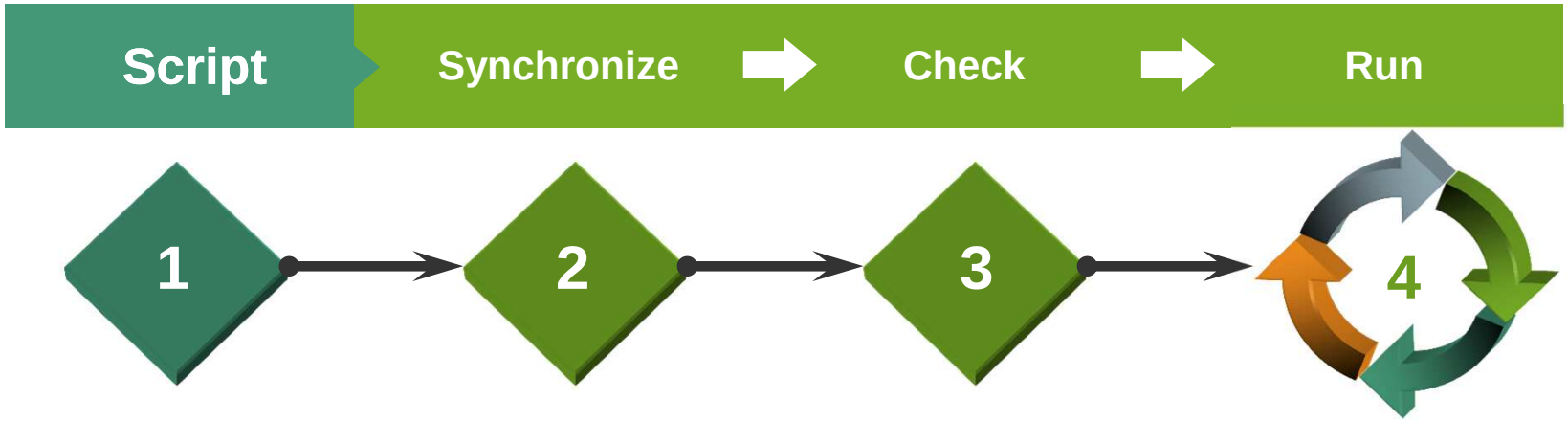
Report

Test Engineers report the test result and move to next test

Check

Test Engineers check the outputs of AUT function

How we do automation test?



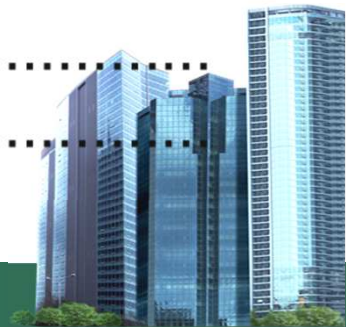
• Test Engineers generate the automated scripts

• Test Engineers synchronize the scripts with the AUT to record output

• Test Engineers define checkpoints and add to the scripts

•Test Engineers run the test scripts for each regression

Notes



Summary

- Test Automation is a method that uses of software to control the execution of tests.
- Commonly, test automation involves automating a manual process currently in.
- Test Automation has benefits such as… However, it still has some pitfalls such as…
- A effective strategy of test automation includes several steps…

Question

Raise your hand, please.

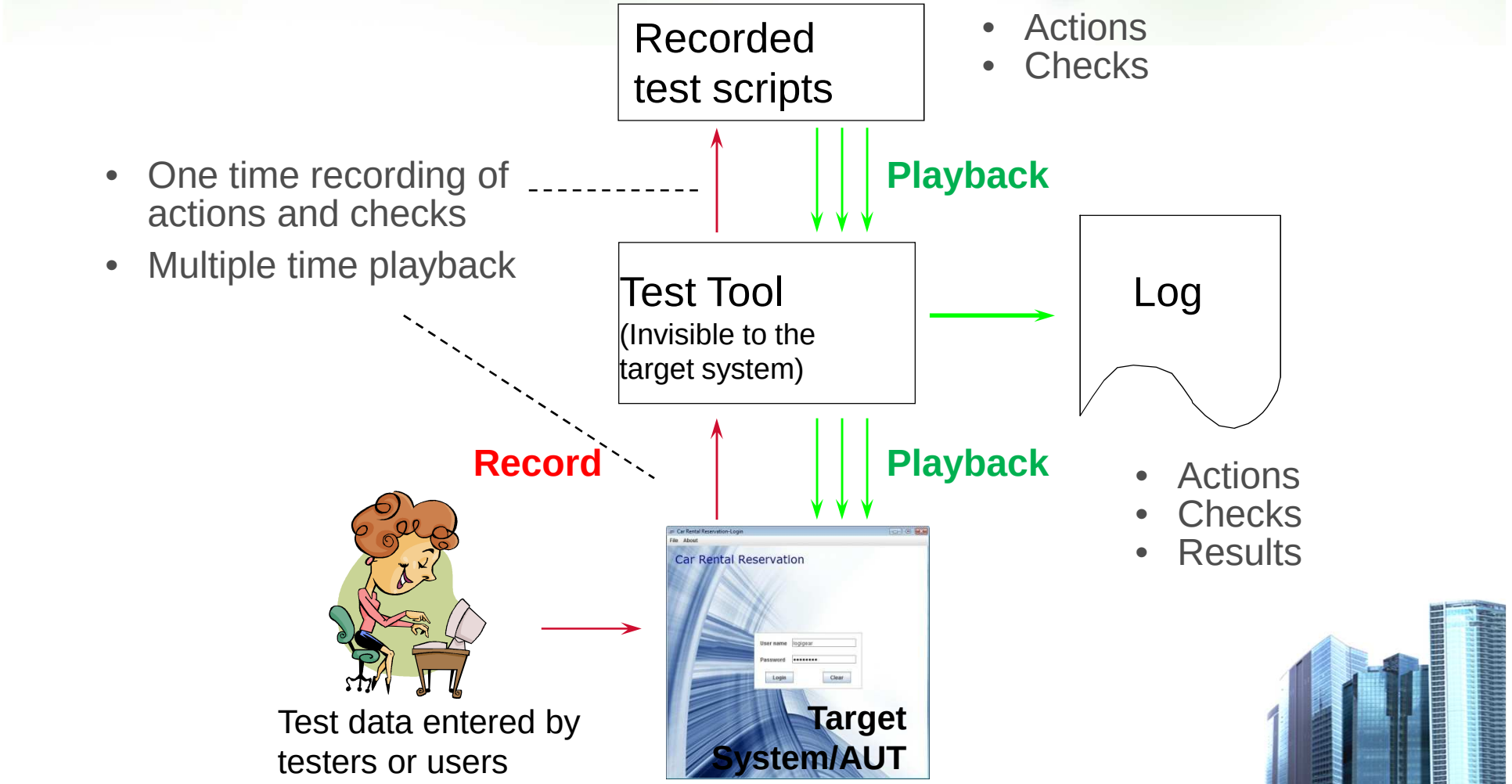
History of Automation Test

- 1st Generation: Record and Playback
- 2nd Generation: Scripting
- 3rd Generation: Data Driven
- 4th Generation: Keyword Driven
  - Keywords:
    - First presented by Hans Buwalda in 1994
    - Dominant in Europe
    - Action Based Testing™

1st Record and Playback

- The automation tool records the steps of the user executing a test, and the tester configures some checks.
- The automation tool generates a script based on the recorded actions.
- The automation tool performs a playback of the recorded test.

How did they do that?



Example: A recorded script

```
select window "Logon"
enter text "username", "administrator"
enter text "password", "testonly"
push button "Ok"
select window "Main"
push button "New Customer"
expect window "Customer Information"
select field "Name"
type "Jones"
select field "First Name"
type "John"
select field "Address"
type "54321 Space Drive"
    .
    .
    .
```

Another example...

WinRunner - [Noname1]

File Edit Create Run Debug Tools Settings Window Help

Verify

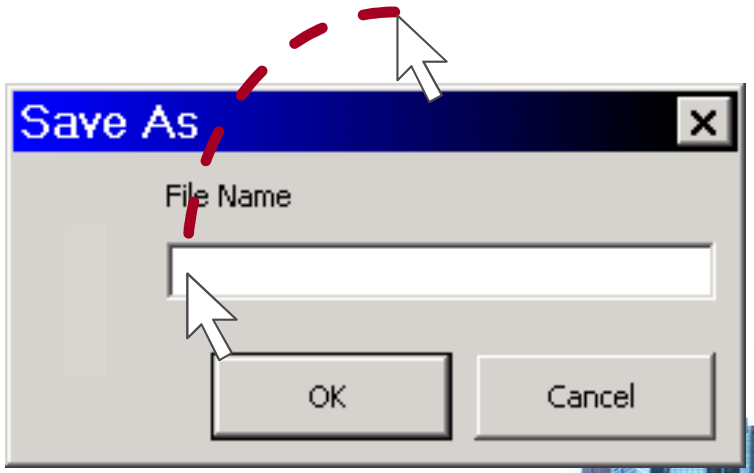
```
# Analog Recording
    move_locator_track (1);
    mtype ("<T55> <kLeft>-<kLeft>+");
    type ("<t3>output14" );
    move_locator_track (2);
    mtype ("<T35><kLeft>-<kLeft>+");
```

mouse movement

mouse click

enter text from keyboard

pause before mouse click



Why did they stop?

- Even small changes to the application under test require manual re-recording of the test.

  ⟶ This first generation is neither efficient nor scalable.

Pros and Cons

- Pros
  - A useful way to learn about the testing tool and automation
- Cons
  - High maintenance cost dealing with changes in the target system of AUT
  - Poor accessibility to test cases
  - Greatly affected by the working system or environment
  - Only suitable for GUI systems

2nd Scripting

- Automation is regarded as a programming activity
  - Parameterize hard-coded values
  - Separate data from code by moving variables to include files
  - Create utility functions to be shared
  - Produce and maintain like any other software
  - Train black-box test engineers to run the scripts

Example: A script

```
Function EnterCustomerJones
    Logon
    Press "New Customer"
    Enter Field "Name", "Jones"
    Enter Field "First Name", "John"
    Enter Field "Address", "54321 Space Drive"
    . . .
    Logoff
End Function

Function OrderProduct
    Logon
    Press "New Order"
    Enter Field "Product", “OurProduct 1.0"
    Enter Field "Amount", "35"
    Enter Field "Delivery", "asap"
    . . .
    LogOff
End Function

. . .
```

Another one...

- A “Silk” test script

```
[-] testcase accessdb()
          [ ] STRING sRecord
          [ ] LIST OF STRING sList

          [ ]// Query the Database using SQL
          [ ] STRING sSqlCommand="SELECT * FROM [Names]"
          [ ]// You have created a DSN connection called MyDataSource for example
          [ ] HANDLE hDataBase=DB_Connect("DSN=MyDataSource")
          [ ] HANDLE hBuffer = DB_ExecuteSql(hDataBase,sSqlCommand)
          [ ]
          [ ] //Print out DataBase
          [-] while (DB_FetchNext(hBuffer,sRecord))
                    [ ] Print(sRecord)
          [ ]
          [ ] DB_FinishSql(hBuffer)
          [ ] DB_Disconnect(hDataBase)
          [ ]
```

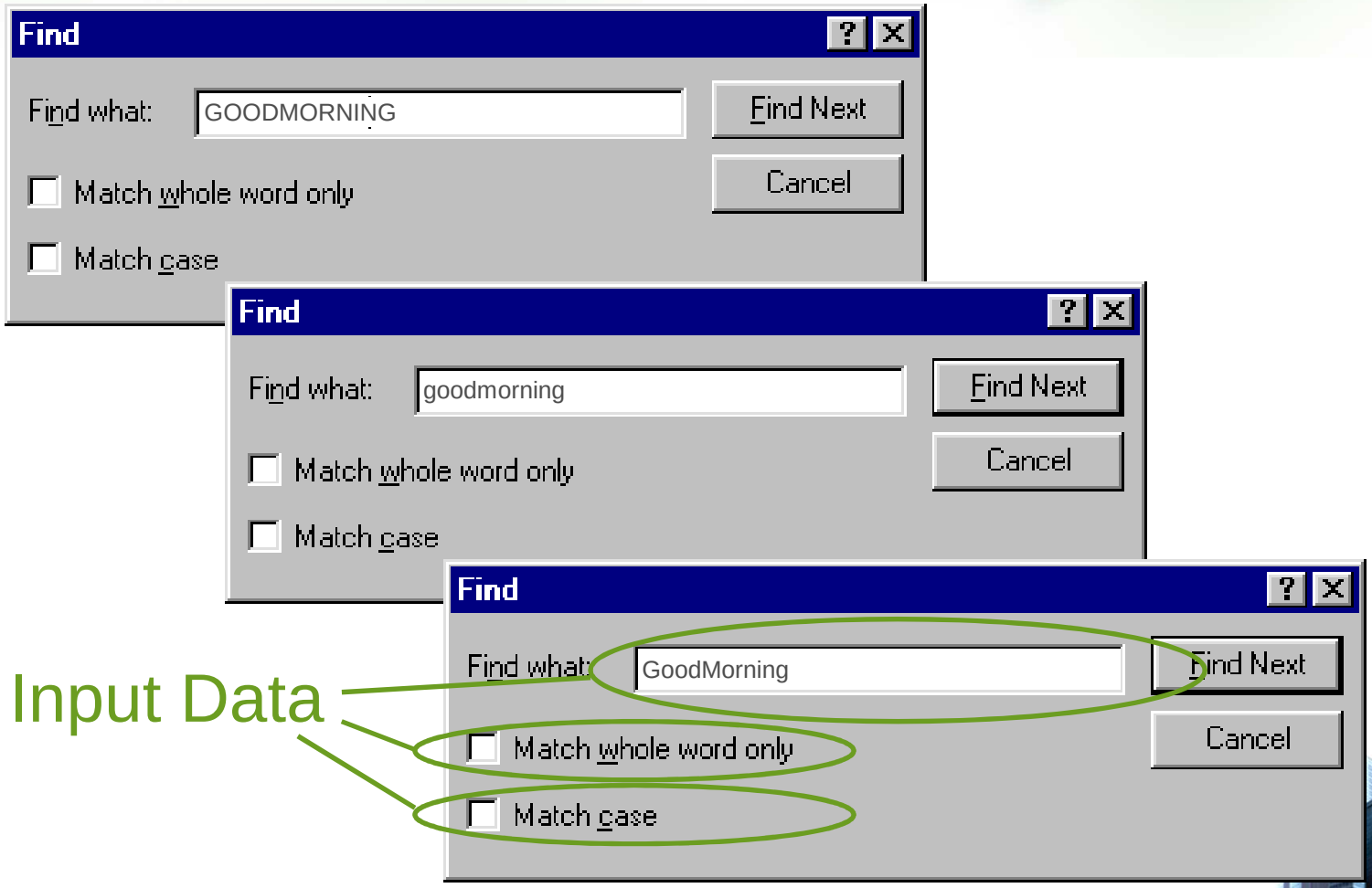
Pros and Cons

- Pros
  - Reusability of scripts for common tasks
  - Improved maintainability
- Cons
  - Poor accessibility to test cases because tests are programmed or scripted
  - Greatly affected by a working system or environment
  - Test implementation must be done by programmers
  - Translation of test cases—3000 tests mean 3000 automated scripts
  - Manageability is still an issue due lack of visibility

3rd Data Driven

- Take advantage of tester's familiarity with test case design and creation using tables and matrices
  - Accommodate localization projects
  - Recognize the importance of patterns in test cases
  - Enable testers to catalog test cases with spreadsheets in software such as Excel
  - Enable testers to specify expected results in spreadsheets

How did they do Data Driven?



How did they do Data Driven?

- File with test data:

nr	case	whole	pattern	matches
1	off	off	GOODMORNING	4
2	off	on	GOODMORNING	3
3	on	off	GOODMORNING	1
4	on	on	GOODMORNING	1
5	off	off	goodmorning	4
6	off	on	goodmorning	3
7	on	off	goodmorning	1
8	on	on	goodmorning	1
9	off	off	GoodMorning	4
10	off	on	GoodMorning	3
11	on	off	GoodMorning	2
12	on	on	GoodMorning	1

- Text file used in the test:

```
goodmorning
GOODMORNING
GoodMorning
GoodMorningVietnam
```

- Data driven script:

```
for each line in the file do
   -open the find dialog
   -read a line from the file
   -use the values to fill dialog
   -press find button
   -check amount of matches
   -close the dialog
```

Pros

- Pros
  - Separation of action and data enables tests to receive combinations of test data
  - Test data can be stored in a separate file or spreadsheet for better maintainability
  - Improved reusability and maintainability

… and Cons

- Cons
  - Actions containing business logic are in automated scripts
  - Maintainability of changes in business logics as well as interface is still poor
  - Poor visibility because accessibility to the tests is still poor
  - Not scalable (e.g., reaching 95% of automation)

So far, the challenges are…

- Too much programming
- Response to rate of changes (software)
- Low reusability/scalability
- Too many platforms/technologies
- Too expensive if the returned value is low

4th Keyword Driven

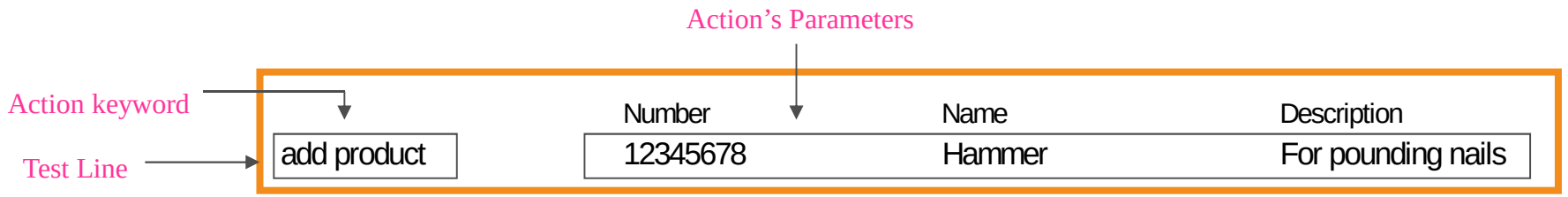
- Tests are broken down into a series of actions representing the business processes of the AUT
- Individual actions are automated, not the tests.
- The tests are stored externally from the test scripts, typically in a spreadsheet.

More about Keyword Driven

- Keyword-based test automation's elegant, structured approach
  - simplifies the work even further
  - reducing the time and expense of test design, automation, and execution
  - by allowing team members to focus on their areas of individual expertise.
- Non-technical testers and business analysts can
  - develop executable test automation from keywords representing recognizable end-user actions.

Action Based Testing (ABT)

- Test Cases are
  - written or described through a series of Test Lines
  - using parameterized and reusable Action keywords



Overcome challenges with ABT

- Separation of actions, interfaces, logic and data
  - Test data can go in a separate file or spreadsheet
  - Test logic can go in a separate file or spreadsheet
  - Interface can go in a separate file or spreadsheet
  - Action can also go in a separate file or spreadsheet
- Action can be reused

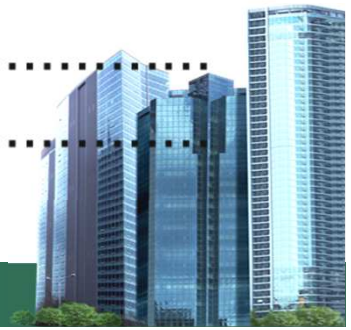
Automation Tools

Automation Tool	HP	IBM	Others
Functional	WinRunner QuickTestPro	Functional Tester Rational Robot	SilkTest TestPartner TestComplete
Performance	LoadRunner AstraLoad Test	Performance Tester	SilkPerformer WebLoad OpenSTA
Management	Test Director Quality Center	Rational TestManager	SilkCentral Test Manager
Memory		Purify	JProbe Profiler

Automation Test Workload

Workload	Test Case Automation	Test Case Production	Test Data Construction / Selection
Record & Playback	N/A	Test Engineer	N/A
Scripting	Automation Engineer	Automation Engineer	N/A
Data-driven	Automation Engineer	Automation Engineer	Test Engineer
Keyword-driven	Automation Engineer	Test Engineer	Test Engineer

Notes



Summary

- There are 4 generation of automation test such as…
- The latest approach, Keyword driven, helps to overcome challenges by…
- There are many popular automation tools. Depend on the AUT, we choose the suitable automation testing tool.

Question

Raise your hand, please.

Automation with Keywords

- Keywords – Action Based Testing
- Overcome challenges with Action based Testing
- The idea of Automation Testing

Objectives

- Understand keyword methodology in automation test
- Understand action based testing methodology
- Understand the idea of automation
- Understand how to implement an automation test project

References

- [ABT] Action Based Testing
  - Hans Buwalda, StickyMinds, 2013

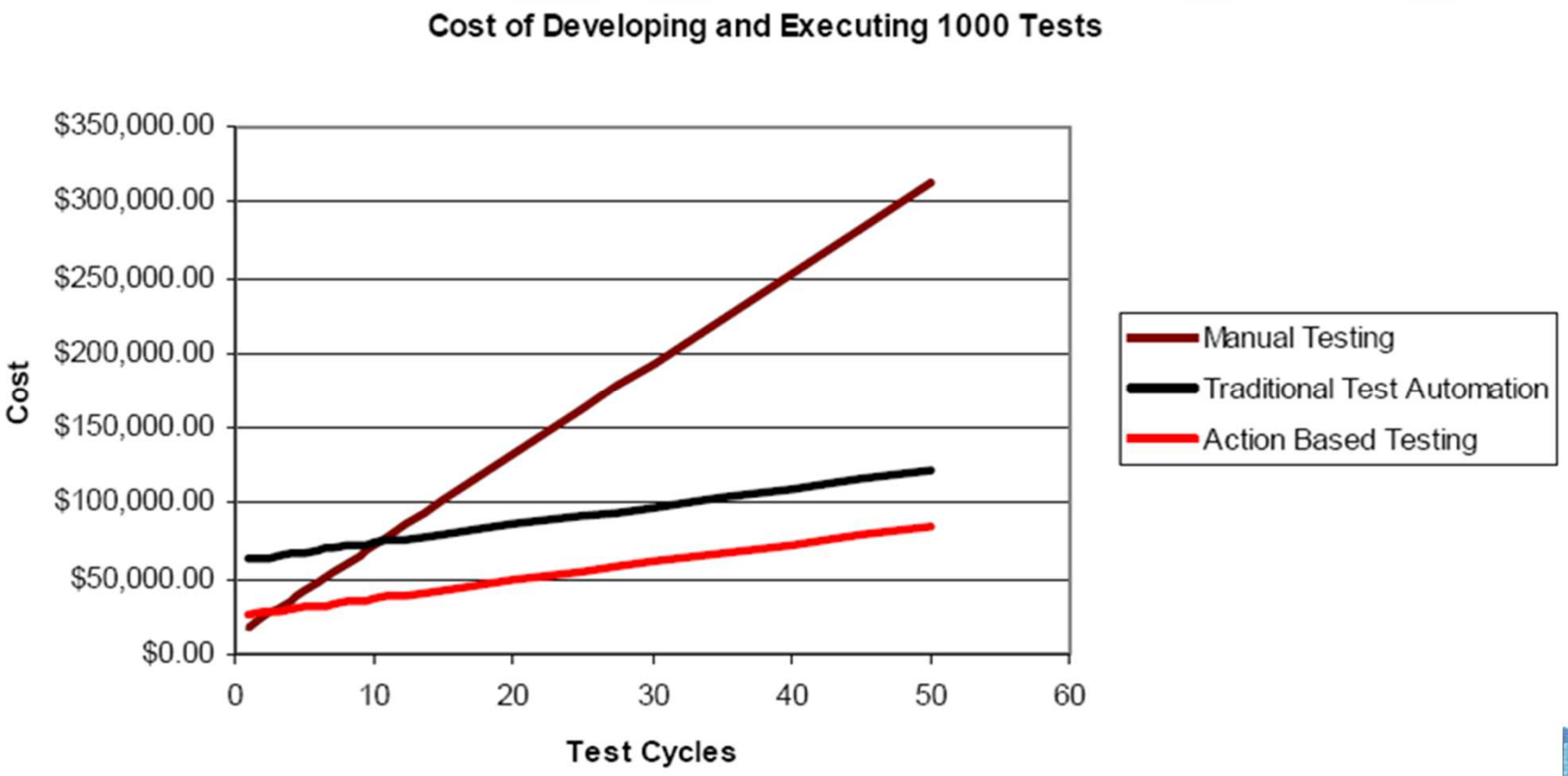
ABT: History

- Action based testing (ABT) is a pioneer among keyword-driven test automation approaches [ABT]
  - Action Based Testing is the proven methodology for software testing and test automation that has been successful in many industries around the world.
  - Hans Buwalda, CTO of LogiGear Corporation, first presented the concept of ABT, a keyword-driven test automation frameworks, to the public in 1994.

Brief overview of ABT

- Action Based Testing
  - saves time and money in test design, test automation and test management,
  - and is a powerful framework for organizing test design, automation and execution around “keywords”

Cost


Cost of Developing and Executing 1000 Tests
$350,000.00
$300,000.00
$250,000.00
$200,000.00
$150,000.00
$100,000.00
$50,000.00
$0.00
Cost
0
10
20
30
40
50
60
Test Cycles
Manual Testing
Traditional Test Automation
Action Based Testing

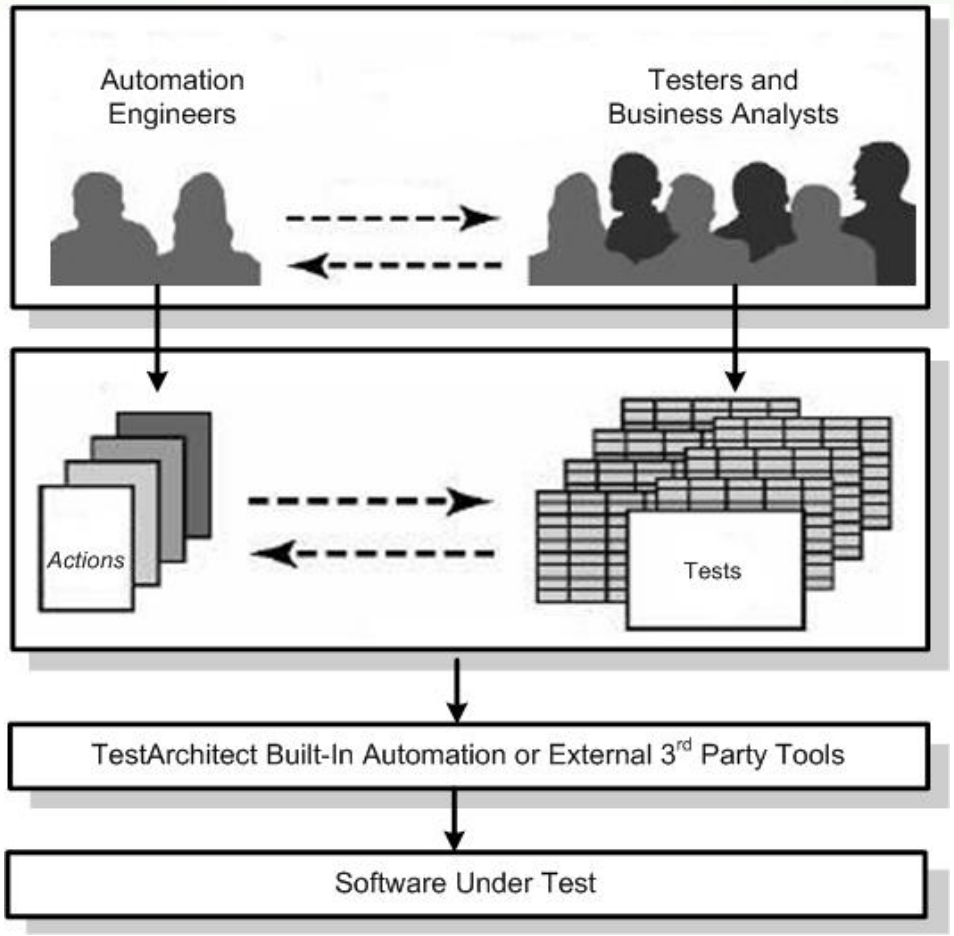
The ABT Framework

People

Practice

Testing

Results



ABT in Practice

- Tests broken down to high-level Actions.
- Testing team build tests from Actions without coding new scripts
- Automation engineers develop and maintain Actions as individual building-blocks that can be combined in any order to design a test.

ABT: an Example

- Example:
  - We have small manual test instruction:
    1. Open IE browser and go to Yahoo Mail http://mail.yahoo.com
    2. Enter yahoo id = jamebond and password = 007007
    3. Click on the 'Sign in' button.
  - Using ABT the same test would be written in a spreadsheet, and typically look like this:

	browser	id	password
login yahoo mail	http://mail.yahoo.com	jamebond	007007

ABT: Key concepts

- Test Module: An individual spreadsheet containing one or more related test cases.

- Test Requirement: An explicit statement of the purpose of a test, based on product requirements or other quality criteria.

ABT: Key concepts

- Test Case: A series of steps and verifications validating one or more test requirements

- Test Line: A single step in a test case, composed of an action and it's parameters

ABT: Actions

- **Actions** are created and communicate with target systems and software applications including the application under test (AUT)
  - through one or more supported Interfaces to carry out intended tasks

    Note:

    For now, think of an Action as a verb or something that you do

ABT: Interfaces

- Information about **Interfaces** such as
  - Windows, Web or Java graphical user interface (GUI)
  - or command-line interface (CLM)

- Interfaces are later defined
  - to enable actions to send inputs to and receive outputs from the target system and/or AUT

The ABT Methodology

Step 1: Testers define the test modules

- Determine how your tests will be divided into different test modules. Some criteria include:
  - Functional area of the SUT
  - Type of test (i.e. Scenario-based tests, requirements validation, boundary checks, etc.)
  - Quality attributes (i.e. performance, reliability, etc.)

The ABT Methodology

Step 2: Testers define the test requirements

- Within each test module, explicitly state the objectives/requirements of the test module. These can be based on:
  - Product/ marketing/ business requirements
  - Functional specifications/ use cases
  - User acceptance criteria
  - Additional quality criteria identified by stake-holders

The ABT Methodology

Step 3: Testers define the test cases

- Within each test module, define test cases which verify the test requirements
- Test cases should be defined as
  - a series of high-level steps representing the high-level business processes of your application,
  - rather than a series of low-level UI interactions
    - For example: ‘login’, rather than ‘enter’, ‘enter’, ‘click’, etc.

The ABT Methodology

Step 4: Automation engineers implement the high-level actions

- Automation engineers will determine
  - how to implement the high-level actions by using a series of lower-level actions and/or programming when necessary
- Automation engineers will help the testers refine their actions to be more reusable
  - Suggest additional parameters, etc.

The advantage of ABT

Visibility

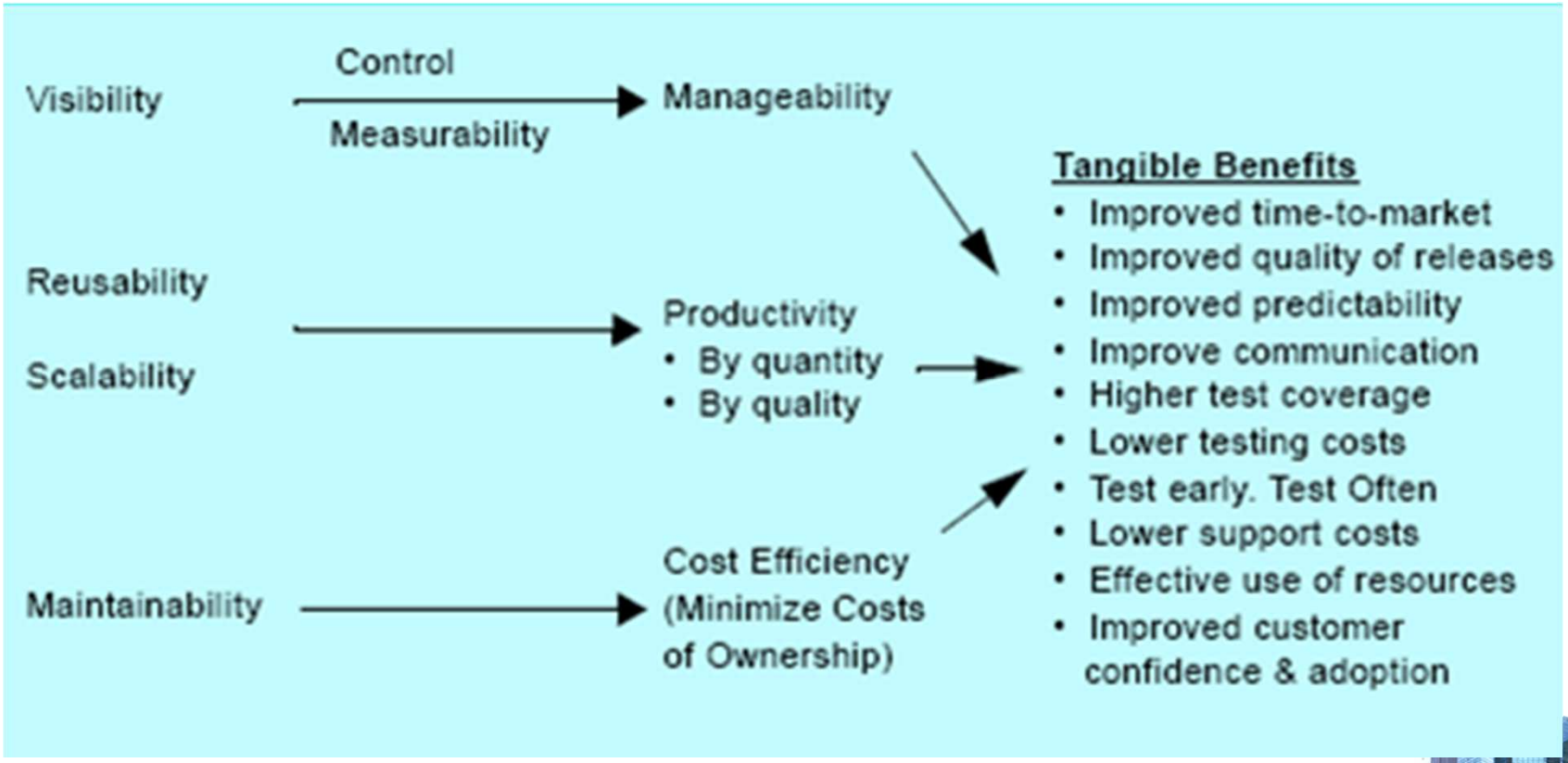
More advantage...

Reusability & Scalability

And more...

Maintainability

And the overall…



ABT: Production Life-Cycle

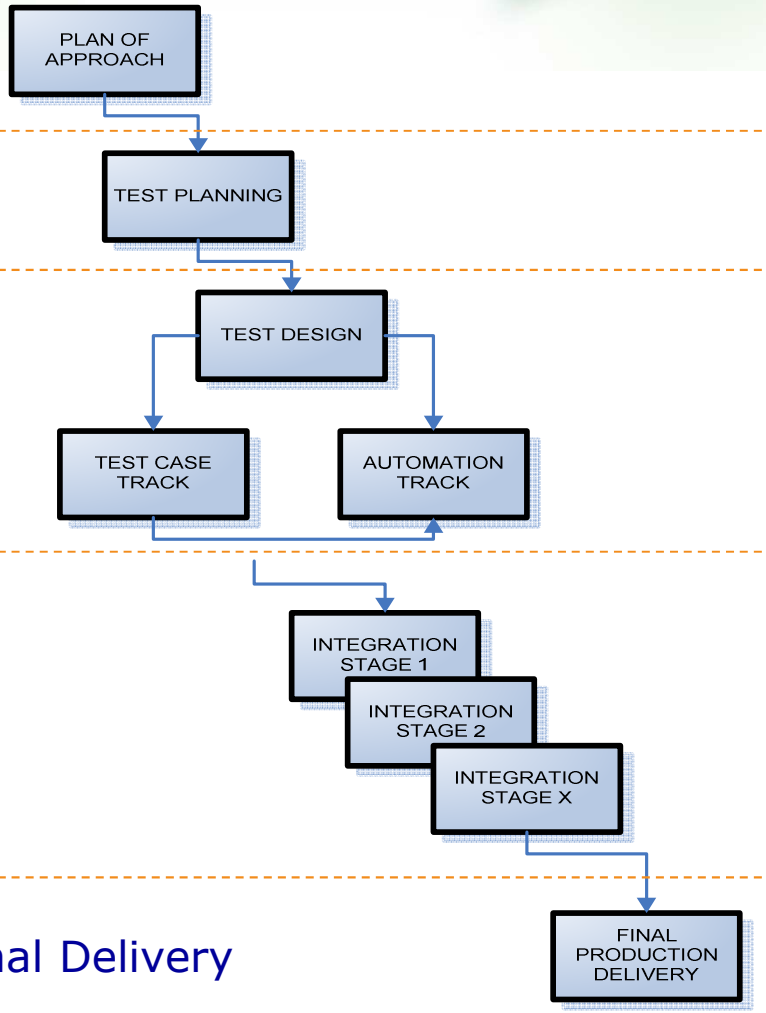
Phase 1: Plan of Approach

Phase 2: Pre-production

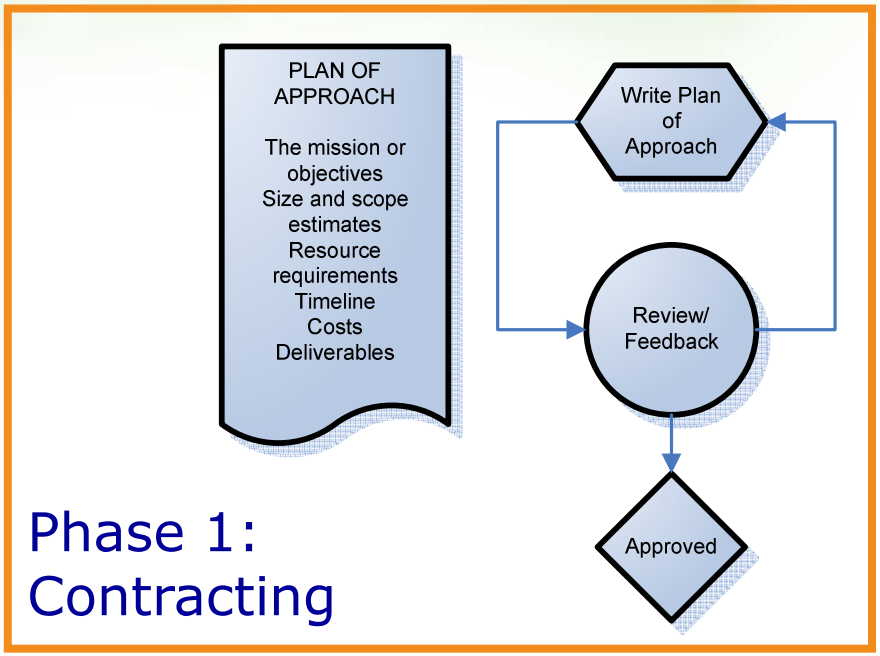
Phase 3: ABT Production

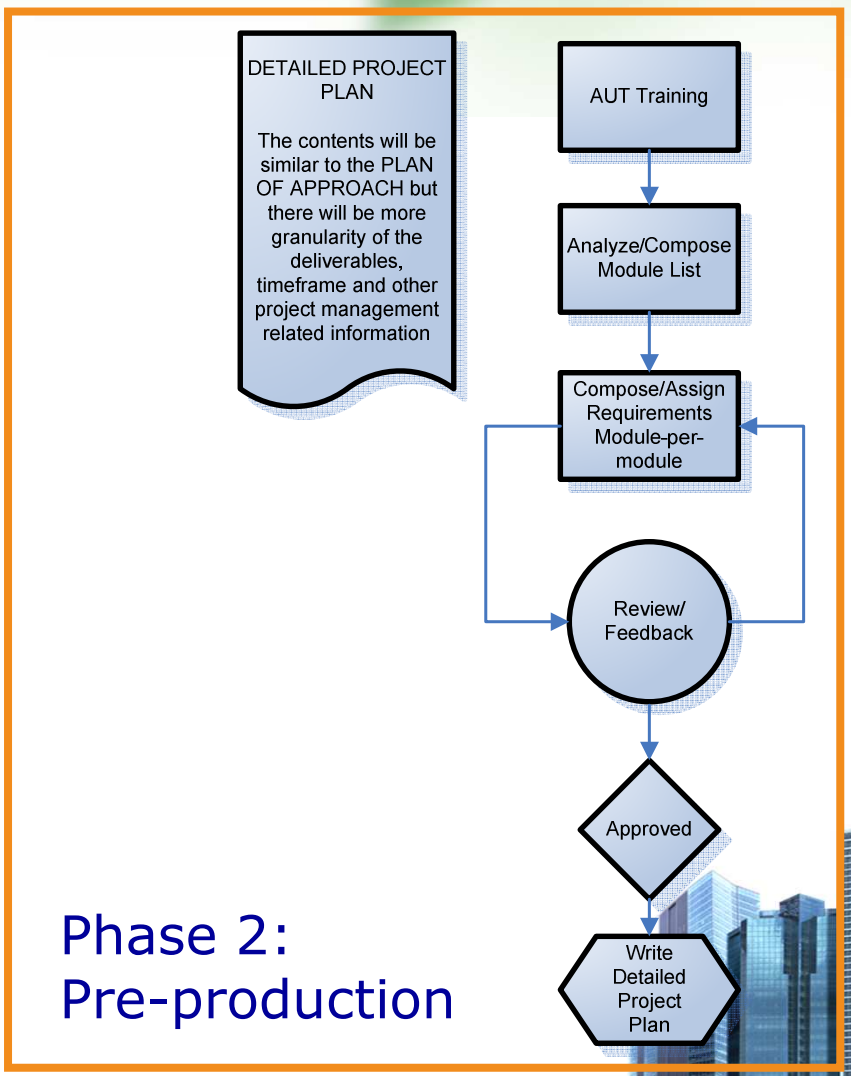
Phase 4: Integration on Test Environment and Others as Required

Phase 5: Post-production Final Delivery

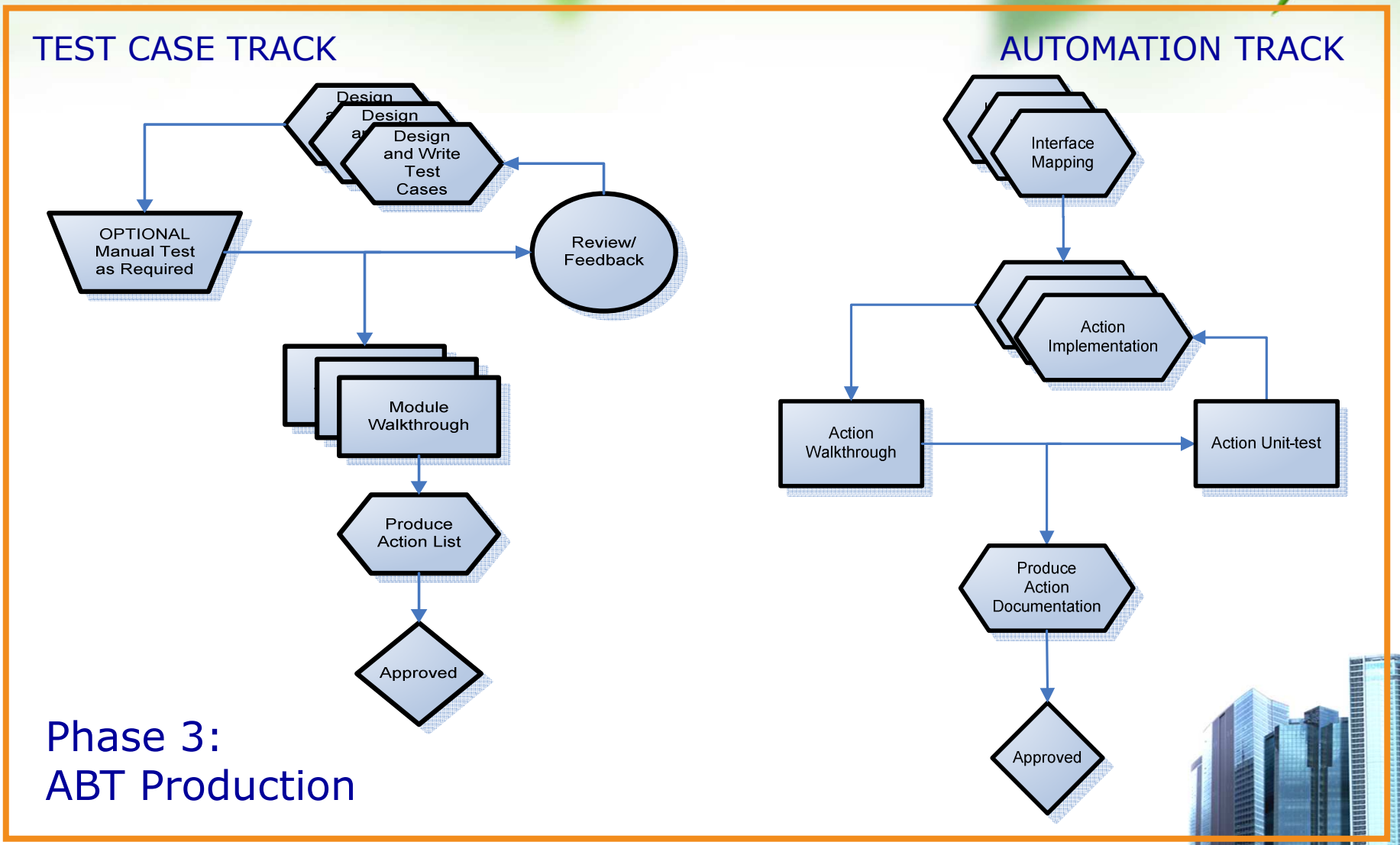


Production Details



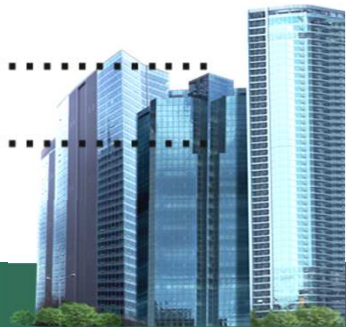


Production Details



Phase 3:
ABT Production

Notes



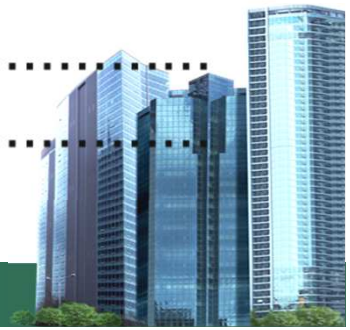
Summary

- Action based testing (ABT), a form of keyword-driven testing
- In an ABT framework, tests are broken down to high-level Actions.
- In general, an ABT Production Life Cycle has 4 phases
- ABT saves time and money

Question

Raise your hand, please.

Notes



The idea of Automation Test

- The general idea
- Searching for objects
- Object Repository, Object Viewer, DataSet
- The model

The general idea

- Perform a test line – an action
  - A primitive action: do it
  - A high-level action: perform each line of it
- Perform next test line…

Perform a primitive action

Primitive actions

- Primitive actions
  - supported by the automation test tools

- 2 types of primitive actions
  - general actions
  - object actions (window actions and control actions)

Object actions

- The common format
  - Action name
  - Window/Control name
  - Arguments

- The general idea
  - Searching for the object (Windows/Control)
  - Perform action on the (Window/Control) using additional arguments

Object criteria

- Criteria
  - A pair of property name and property value

- To identify an object, we need a set of criteria.

Optimize object criteria set

- Object criteria set is optimized if
  - It can be used to identify object uniquely
  - The number of criteria is minimum

- The automation test tool will suggest the optimization of object criteria set.

Searching for objects

- Go through all objects, for each one
  - Obtain properties of the object
  - Matching object properties with the criteria set
  - If its properties are matched, we found the object
  - If not, go to next object

The scope of searching

- "Go through all object"
  - How many objects?
  - Cannot have a criteria set?

- The solution is: Narrow the scope

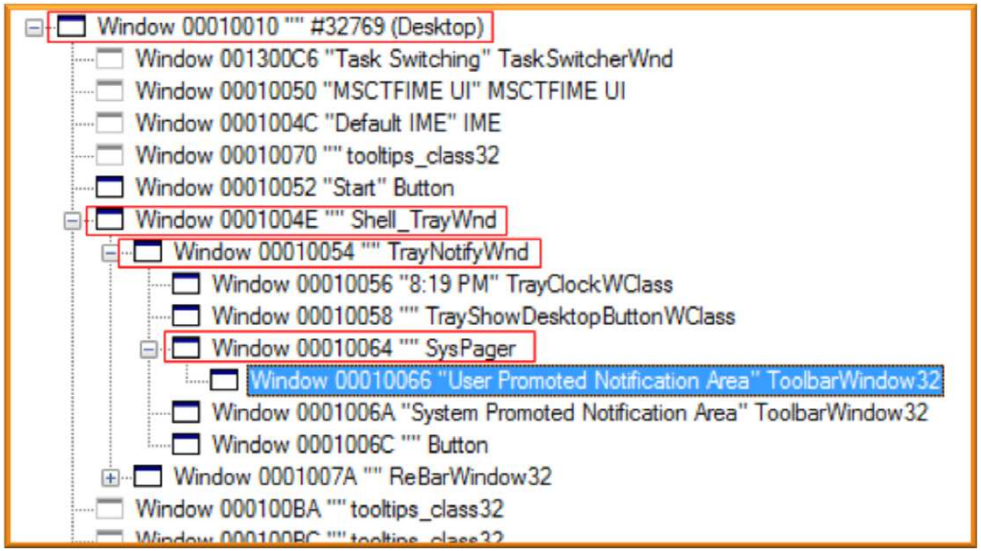
Scopes

- The scope of searching objects could be
  - The containing window
  - The containing object
    - Tab page
    - Panel
    - Other parental control

Recursive Searching

Recursive Searching for Object

- If the object has parent
  - recursive searching for parent object
- Once parent object has been found
  - searching for object within parent object



Object repository

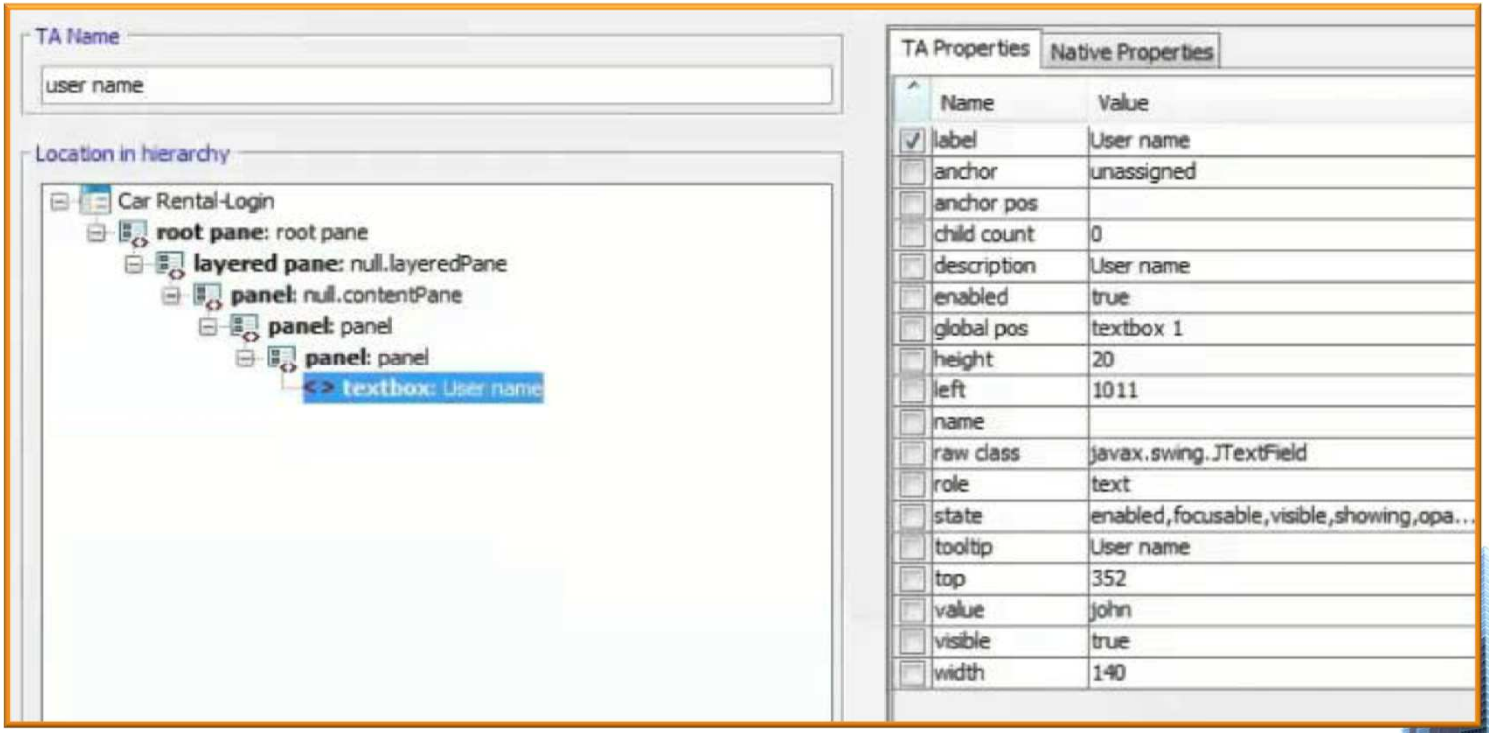
- Object repository
  - A place to store objects' definitions

- Within the repository
  - An object has a unique name
  - An object is described by a set of criteria

btnOk	Parent	**HelloWin**
	Caption	"Ok"
	Class	Button

Object Viewer

- A tool provided by Automation test tool
  - Obtain object in hierarchy UI tree
  - Obtain object's properties to optimized criteria set

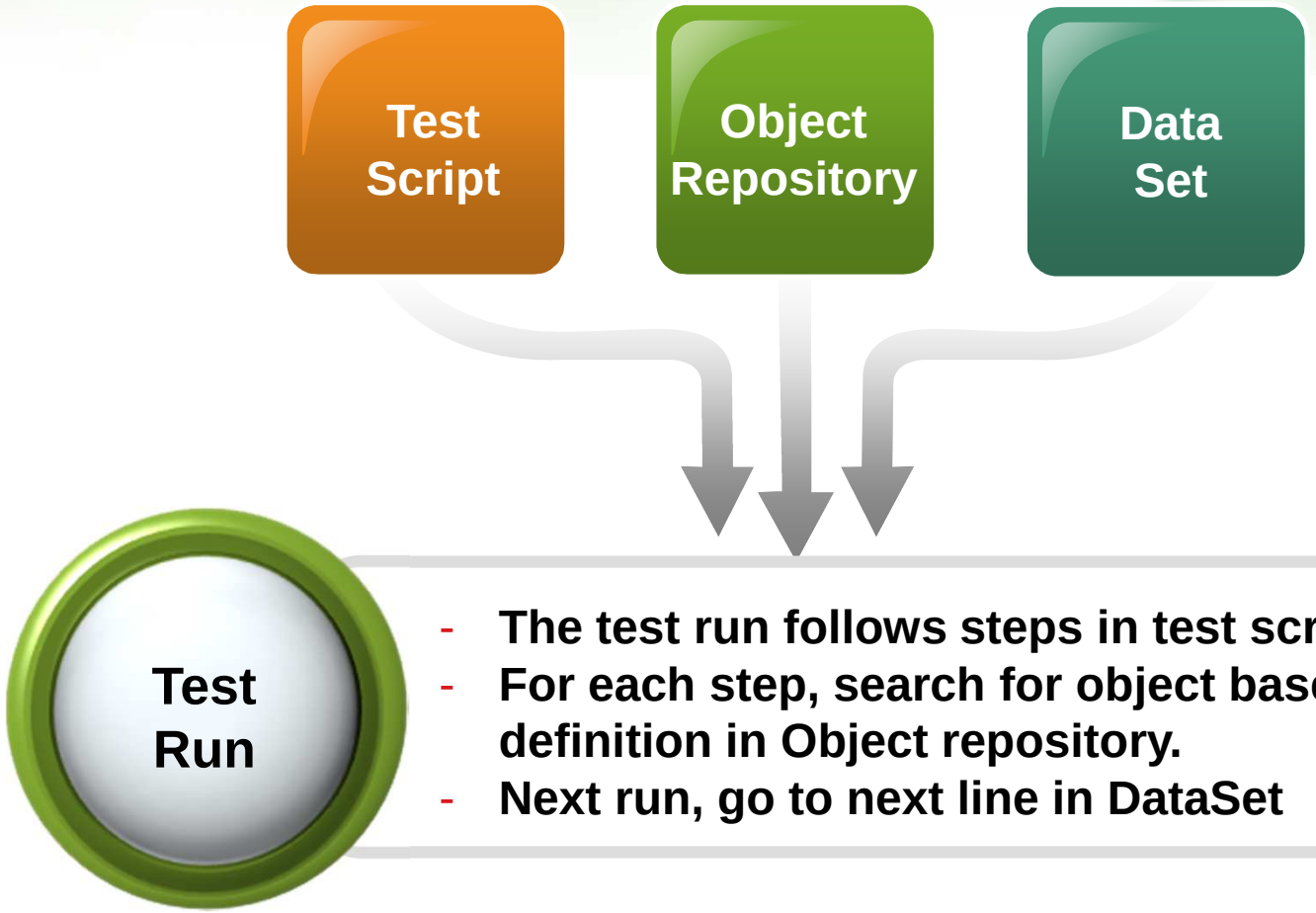


DataSet

- A separated file for storing data to test
- A DataSet is in the form of a table
  - Columns represent for “fields”
  - Rows represent for a data to test

DataSet: Register User				
UserID	**Password**	**Gender**	**Age**	**Country**
Jame	007	Male	40	UK
Sarah	p@ss	Female	26	US

The model



- The test run follows steps in test script.
- For each step, search for object based on definition in Object repository.
- Next run, go to next line in DataSet

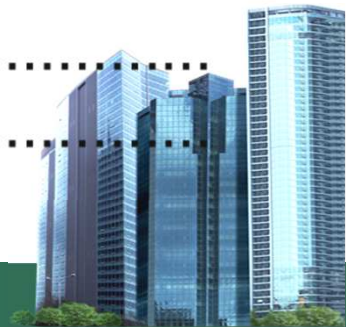
Summary

- The general idea is searching
- Searching object based on definitions
- Object Repository is used to store objects' definitions
- Object Viewer is used to explore objects' properties

Question

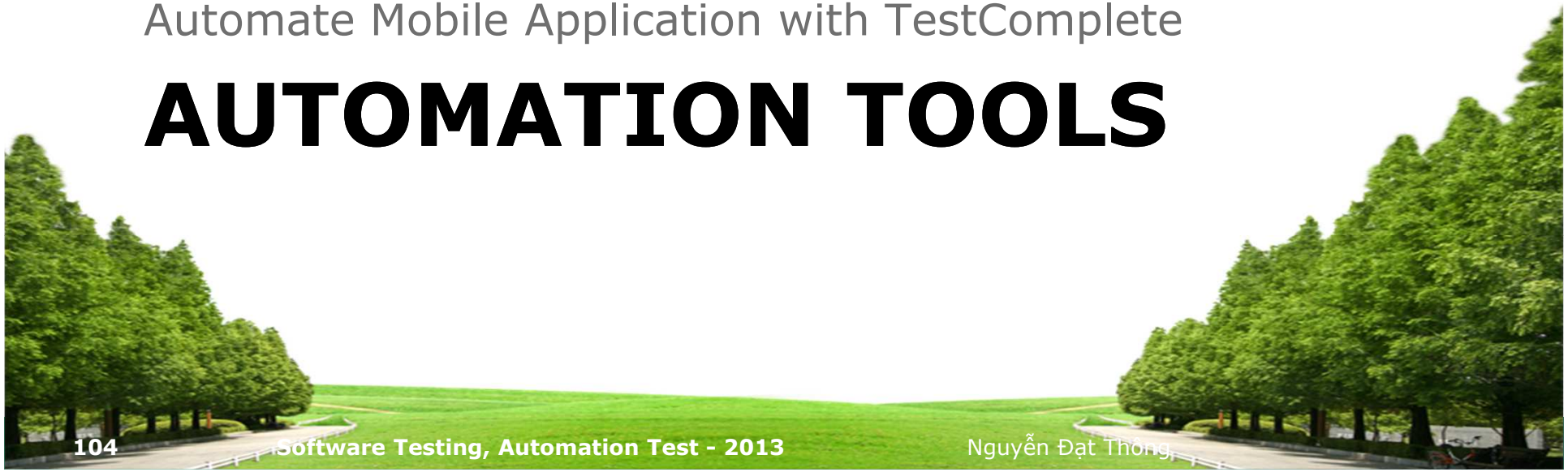
Raise your hand, please.

Notes



Automate Web Applications with Selenium
Automate Desktop Applications with TestArchitect™
Automate Mobile Application with TestComplete

AUTOMATION TOOLS



Selenium

- Selenium IDE: Record and Playback
- Selenese: the actions
  - Object definitions?
- Reusability, Scalability and Maintainability
  - **Frameworks**

TestArchitect™

- Object Viewer and Interface (Object repository)
- Actions and High-level actions
- DataSet

TestComplete

Question

Raise your hand, please.

Notes