



Strategies and Tactics for Global Test Automation

Software Test & Performance Conference 2006
The Hyatt Regency Cambridge, Boston, MA

Hung Q. Nguyen
LogiGear® Corporation
November 7 - 9, 2006

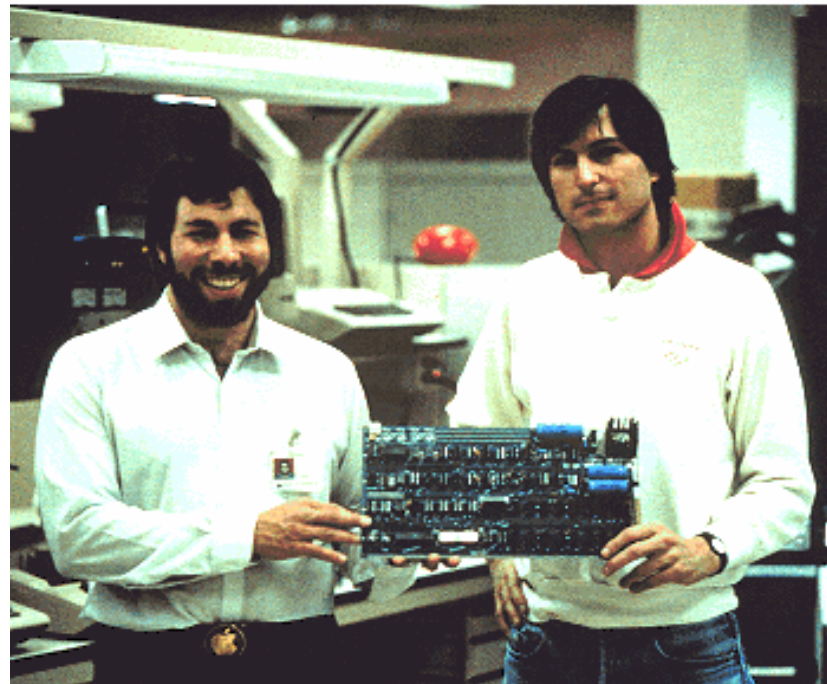


Global Test Automation

Imagine the Possibilities!

Remember Them?

From Computer Desktop Encyclopedia
Reproduced with permission.
© 1996 Apple Computer, Inc.



The Two Steves

Wozniak and Jobs (left to right) pioneered the microcomputer revolution. Wozniak's engineering and Job's charisma truly built a legend. Here they hold the motherboard from the Apple I, Apple's first computer. *(Image courtesy of Apple Computer, Inc.)*



Testing for Executives

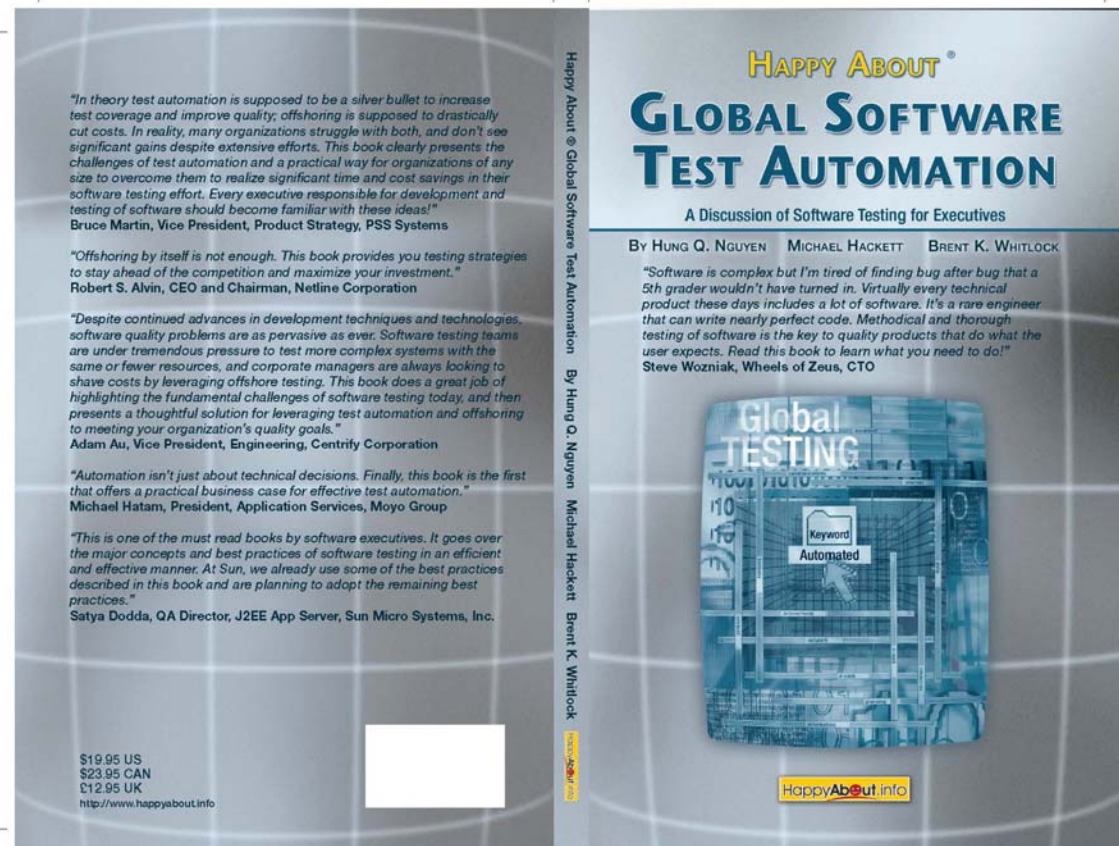
Released July 2006

As authors, we're honored to be endorsed.

"...Methodical and thorough testing of software is the key to quality products that do what the user expects.

Read this book to learn what you need to do!"

**Steve Wozniak, CTO
Wheels of Zeus**





About the Book

This is the ***first*** book to offer software testing strategies and tactics for executives. Written by executives and endorsed by executives, it is also the ***first*** to offer a practical business case for effective test automation, as part of the innovative new approach to software testing:

Global Test Automation—a proven solution, backed by case studies, that leverages both test automation and offshoring to meet your organization's quality goals.



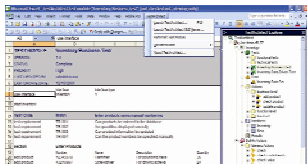

What is Global Test Automation?

Global Test Automation (GTA) is an integration of the latest test automation methodologies and technologies with global resource strategies to fully capitalize on the speed and cost advantages of best practices in automation and global sourcing.

The Challenge

1. Software testing is necessary.
2. Software testing takes time.
3. Software testing costs money.

↑
SAVE TIME

 Test Automation	How can we save both time <i>and</i> money?
Manual Testing	 Offshore Resource

→
SAVE MONEY

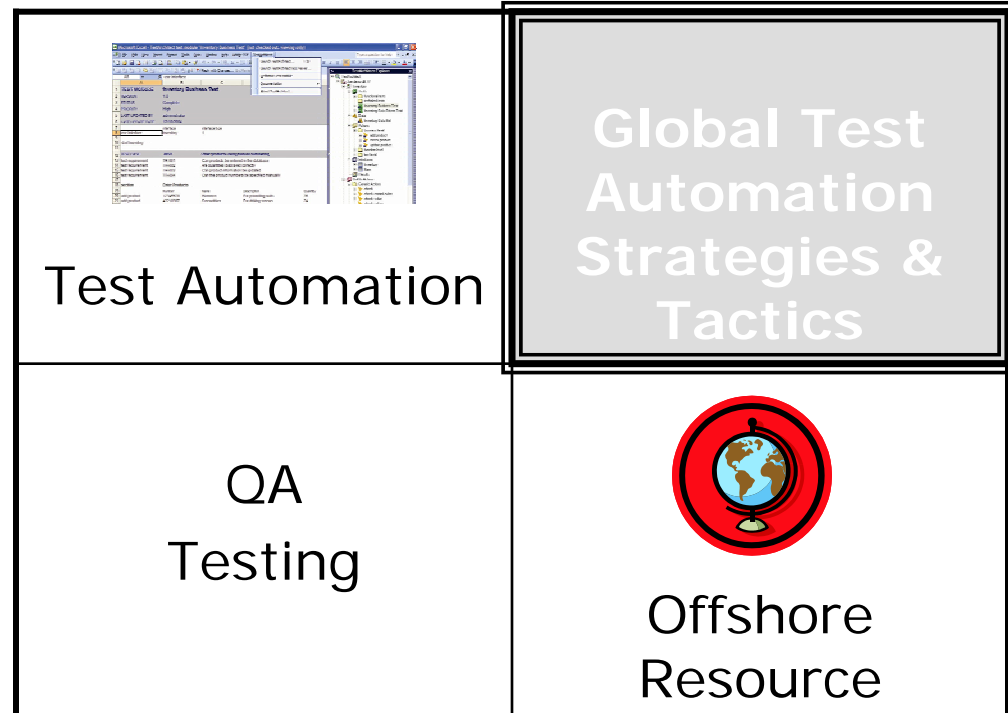
The Objective

We want to be here!



1. High Speed.
2. Low Cost.
3. Uncompromised Quality.

↑
SAVE TIME



→
SAVE MONEY



Global Test Automation

But first,...
...the challenges



Top 5 Offshore Testing Challenges

1. Communication
2. Skill sets
3. Management issues
4. Vendor and infrastructure issues
5. Productivity and security risk



Offshore Resource Challenges

Communication—Difficulty of management

– Cultural

- Non-responsive
- Non-communicative

– Time zone

- Slow speed of communication
- Interminable workday

– Language barrier

– Poor management of two-way communication and expectation



Outsource Survey

How likely would you be to drop a service provider if they use offshore service reps with poor English skills? (345 responses)

Not likely: 9%

Somewhat likely: 9%

Very likely: 39%

Done deal, they're dropped!: 43%

Source: Information Week's Outsourcing Pipeline newsletter



Offshore Resource Challenges

Skill sets—Poor productivity and quality of service

- Lack of QA/test competency ⁽¹⁾
 - Test design skills
 - Bug finding skills
 - Bug reporting skills
 - Bug analysis/isolation skills
- Domain expertise
 - Knowledge of product under test ⁽¹⁾
 - Domain knowledge in the category of the product under test ⁽²⁾
- Technical knowledge ⁽¹⁾
- Others (Varies among regions) ⁽²⁾
 - Lack of versatility or not multitasking-aware
 - Lack of systematic approach
 - Lack of creativity
- Skill mismatch—Want to code; don't want to test! ⁽¹⁾

⁽¹⁾ Not unique but the problem is exacerbated

⁽²⁾ Unique



Offshore Resource Challenges

Management issues

- Lack of visibility and control over the business process and distributed test teams
- Poor management of test products or artifacts
 - Share and control the integrity of test artifacts including test requirements, test designs and test scripts
 - Share test results
- Lack of a test management platform and methodology to:
 - Track and manage the work done by the remote team
 - Easily generate progress report metrics to control testing activities, schedules and risk



Offshore Resource Challenges

Management issues (cont.)

- Big management drain on US staff
- Lack of trust
- Savings not worth the hassle!
 - It's not a one-to-one saving
 - Volume does not reach critical mass to achieve meaningful savings
- No clearly set goals
 - Lack of measurements
 - Lack of analysis to evaluate effectiveness
 - Lack of improvement initiatives
- Inadequate preparation and training for both US and offshore staff



Outsource Survey

US companies have not properly anticipated and planned for how much oversight and management is needed to make offshoring positive.

The savings gained through offshore outsourcing are worth the additional management challenges.

(118 responses)

Strongly Agree: 14%

Agree: 17%

Unsure: 11%

Disagree: 26%

Strongly Disagree: 31%

Source: Information Week's Outsourcing Pipeline newsletter



Offshore Resource Challenges

Vendor and infrastructure issues

- Few best-of-breed offshore testing vendors
 - There are many all-in-one outsourcing shops but few are specialized in software testing with credible expertise and a track record.
 - Development and testing activities are not the same. Many shops try to do everything, and one can't be expert on everything.
 - Many offshore shops take pride on being CMM-certified
 - Being CMM certified is a plus but does not reflect strong testing capability. Good testers are required to have strong testing skills, domain expertise, technical and communications. Acquiring the right skill sets is essential.

LogiGear® Offshore Resource Challenges

Vendor and infrastructure issues (cont.)

- Poor bandwidth
 - Infrastructure, from uninterrupted power supply to network bandwidth is essential for any offshore facility.
 - There have been numerous stories of builds taking so long to transfer to an offshore facility due to lack of bandwidth that time efficiencies have been lost.
- Service disruptions due to poor back-up plans for outages.
- Attrition problem drives higher cost.
 - Attrition is a growing problem, particularly at the middle manager level.
 - Staff turnover is/can significantly set back any project and team.

LogiGear® Offshore Resource Challenges

- Productivity and security risks
 - Time to productivity
 - Team integration takes time
 - Ramping up takes time
 - Training takes time
 - Security
 - Protection of intellectual property
 - Protection of confidentiality
 - Protection of physical property



Top 5 Test Automation Challenges

1. Uncertainty—Test automation is a risky investment.
2. Scalability and maintainability of test automation—Test automation is a costly investment.
3. Management of test automation production (Manageability and control) is a big problem.
4. Technology vs. people issues.
5. Poor methods used.



Test Automation Challenges

Uncertainty—Test automation is a risky investment

- Companies know that they need to automate as much as possible but aren't sure how to make it successful.
- Companies want to achieve high-degree of automation without incurring high total cost of ownership but aren't sure how to make it happen.
- The uncertainty of how to effectively implement automation coupled with the wishful thinking of a plug & play tool normally leads to an ineffective plan. Cost savings are not realized.



Test Automation Challenges

Scalability and maintainability of test automation—Test automation is a costly investment.

A simplified method for calculating the cost of test automation:

Cost of test automation

- = Cost of test tool
- + Labor costs of script creation
- + Labor costs of script maintenance

Labor costs of script creation is much more difficult to quantify because it depends on

- The automation framework and methodology
- Having the right balance of people who design tests (good testers may not necessarily have good programming skills) and people who create automated test scripts (good programmers don't necessarily have good test domain expertise and test design skills)
- How likely the created test scripts will break when the application under test changes.



Test Automation Challenges

Manageability and control of test automation production

- Most automation tools lack an easy reporting mechanism with visibility into what automation is doing, what tests are being written, run, how often and what are the results.
- Testing and test automation is a process. To successfully manage a process, you need to be able to quantitatively and qualitatively assess the activities and progress throughout the testing lifecycle. You need a well designed yet simple process, as well as a tool that offers the appropriate metrics to bring visibility of testing and test automation activities.
- Management of technical teams. Many test leads lack the technical skill sets to both provide technical leadership to test developers and solid proficiency in the process to manage and control production. Test Leads trained in managing manual test projects need different skills, not only technical but management to run and communicate an automated test effort.



Test Automation Challenges

- Technology vs. People issues

The tool will not solve the problem—Automated testing is software development. It needs process, procedures, design, architecture, standards, conventions, reviews. Few organizations have the availability of resources to commit to this type of effort.

 - No good tool for all of your needs
 - The promises of what automation tools can do, particularly in the self-running nature of test creation is grossly misleading.
 - Even some of the better automation tools have bad communication features, with reporting mechanisms that need the purchase of other tools to clean up their lack in reporting. Therefore the automation and communication need different tools.
 - Tool is expensive comparing to benefit it delivers
 - With limited benefit and high maintenance costs substantial benefit from automation programs is rarely realized. This would be bad for business as is but limited success is better than tools winding up as shelfware, which is the most common result with a total loss of benefit. With such limited benefit, the cost of the tools is never recovered.



Test Automation Challenges

- Technology vs. people issues (cont.)
Test engineers are not using the tool:
 - Lack of training or fear of new technology,
 - Many companies use external, pre-trained teams or individuals.
 - Many teams never get the opportunity to implement an automation program no matter how well designed because of the time investment needed and the adverse effect this would have on current projects.
 - With the high maintenance nature of most automation programs there is often a defeatist attitude on automation teams.



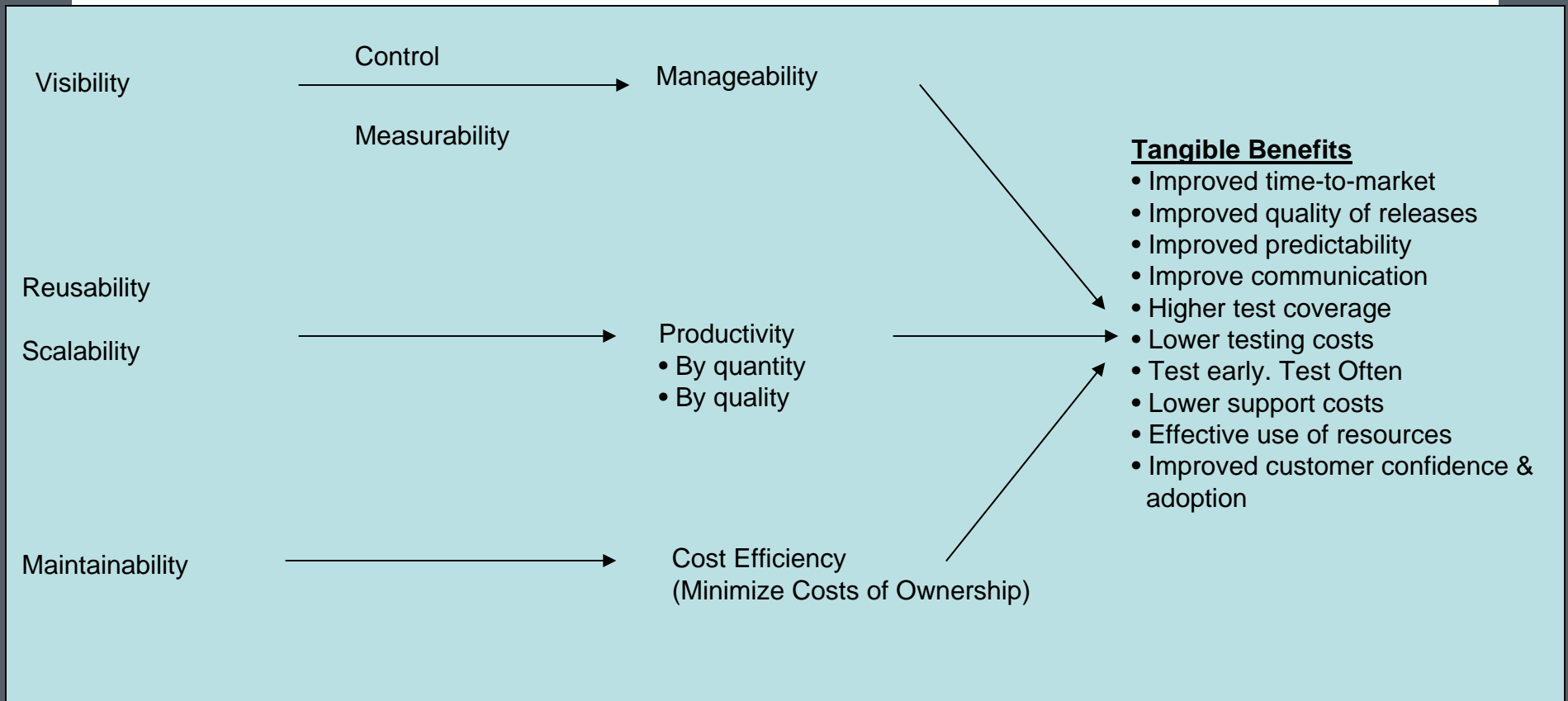
Test Automation Challenges

Poor methods

- An unscalable automaton framework or architecture such as “record and playback” as well as high-maintenance scripted test automation leads to poor reusability.
- Software Development cycles are so fast the team has no time to effectively automate, especially not reusable scripts.



Outflows of Test Automation





More on Global Test Automation

- Global Test Automation (GTA) is an example of an application based on ABT
- ABT enables test engineers (domain experts who may not be skilled in coding) to focus on developing executable tests based on action keywords, while automation engineers (highly skilled technically but who may not be good at developing effective tests) to focus on developing the low-level scripts that implement the keyword-based actions used by the test experts
- This seamlessly makes a good recipe for the best utilization of the skillsets of your staff back home as well as offshore
- GTA is about team-based test automation
- GTA is test-design-centric, not tool-centric



Strategies and Tactics

Developing a Global Test Automation strategy and roadmap focusing on

- Assessment
- Process-driven test strategy
- Leveraging automation
- The method, not the tool
- Selecting the right toolset
- Securing and developing competency
- Visibility—Measure, analyze and optimize



1. Assessing Needs

What is a test assessment?

A test assessment is a data gathering process.

To make decisions we need data.



The Purpose of an Assessment

The goal of doing a test process assessment is to get a clear picture of the testing needs, what is going on in testing, the good things, the problems, possible paths to improvement and strategy development.



Important Note

While the focus is on testing, this effort is much larger than only the test team.

Issues will arise over:

- Who owns quality?

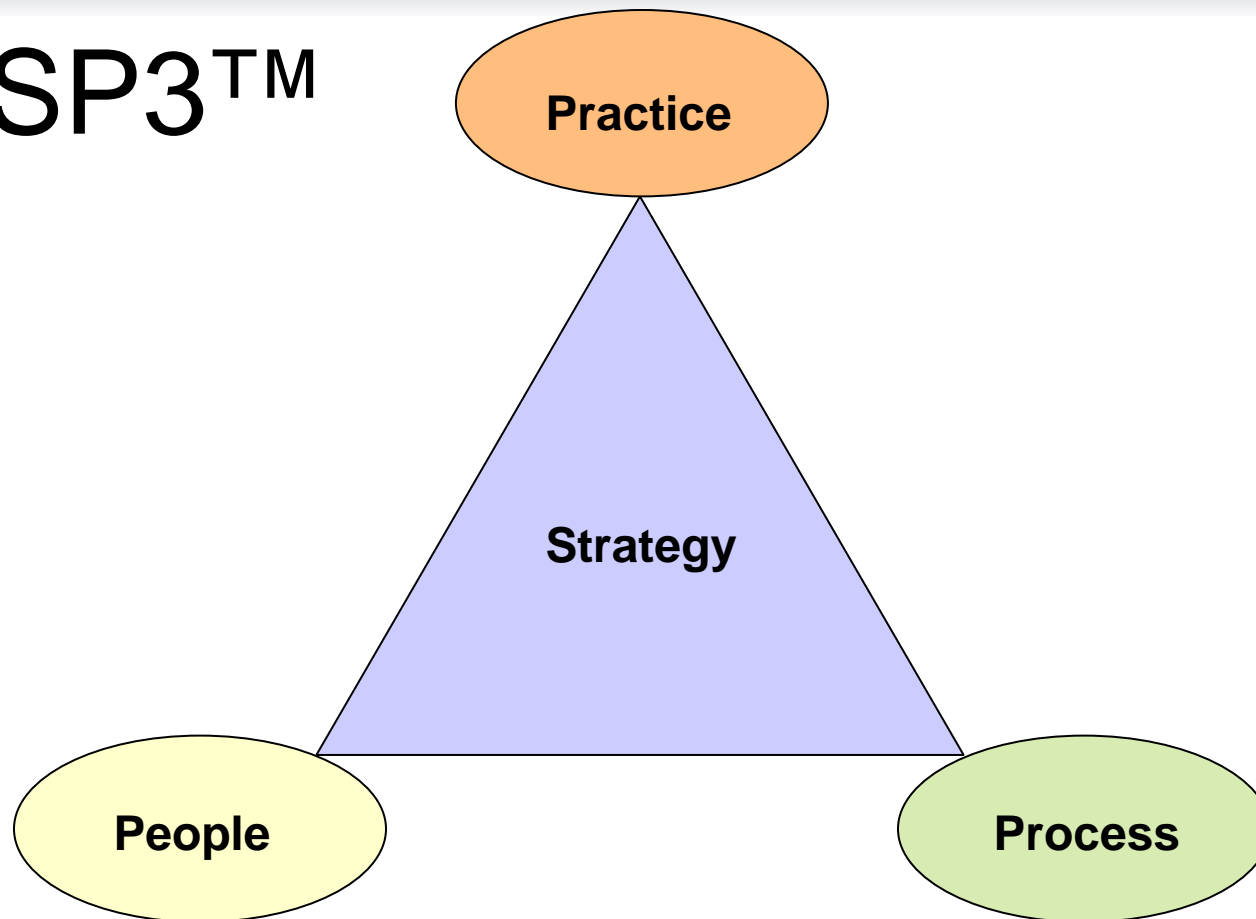
- What is the goal of testing?

If this assessment is done well you may step on some toes!



Testing Strategy in Context

SP3™





Examples of Possible Improvements

- ☑: Yes
- ▣: Maybe
- : No

SP3: Strategy that consists of People, Process and Practice

Process: Life cycle

People: Skill sets, communication and morale

Practice: **Methods and tools**

Plausible Corrective Actions	Faster	Better	Cheaper	SP3
Automate test design	☑	▣	▣	Practice
Automate test documentation	☑	▣	▣	Practice
Automate test execution	☑	▣	▣	Practice
Deploy test automation strategy	☑	▣	▣	Practice
Focus testing on bug-finding rather than documentation	☑	▣	●	Process
Improve meaningful metrics with well defined correlation/corrective action	☑	☑	▣	Practice
Improve visibility of testing/QA activities	☑	☑	☑	Practice
Lessen number of cycles	☑	●	▣	Process
Reduce automation test script maintenance	☑	●	☑	Practice
Reduce test case maintenance	☑	●	☑	Practice
Remove redundancies	☑	●	☑	Practice
Shorten the test cycle	☑	●	▣	Process
Test earlier	☑	▣	●	Process
Upgrade talent through training and/or churning	☑	☑	▣	People
Improve test selection	☑	▣	▣	Practice
Leverage outsourcing including global resources	▣	●	☑	Practice
Broaden test coverage	●	☑	●	Practice
Improve test design	●	☑	●	Practice



A Common Theme of Findings

Your test teams might be doing things right,
but not necessarily the right things!



Test Assessment Ingredients

This assessment should produce a report that reveals the feedback on the maturity of the QA/Testing group as a whole, which includes evaluations of

- QA/Testing human capital,
- processes,
- test strategy,
- methods and tools,
- project scheduling and
- the overall effectiveness.



Test Assessment Ingredients

- The evaluation process should be based on the review and analysis of
 - The current quality related documentation
 - Surveys and interviews
 - Knowledge of best practices
- The content of the report should include:
 - Data collected through surveys and interview sessions.
 - Discoveries: Information that offers feedback on the current state-of-the practice contrasting with standard processes such TPI (Test Process Improvement), TMM (Test Maturity Model) including
 - A Process Scorecard: A metric-based report card for the overall test process assessment.
 - Staff Competency Scorecard: A metric-based report card for the overall human capital assessment.



Test Assessment Ingredients

More on the content of the report

- A Roadmap—A plan consisting of recommendations for improvement.
- Implementation activities.
- Implementation plan and execution.



Test Assessment Ingredients

Assessment methodology

– Phase I - Intake

- Become familiar with the testing organization.
- Identify the full scope of the project.
- Determine which individuals within the organization should be involved.
- Determine what quality criteria should be assessed and what questions to ask of each person involved.
- Determine what documentation to be gathered from the organization for assessment.



Test Assessment Ingredients

- Assessment methodology
 - Phase II – Data Collection
 - Surveys given to personnel involved in the testing and development effort.
 - Face-to-face interviews with personnel involved in the testing and development effort.
 - Assessment of testing environments, office environments, and testing processes.
 - Review of existing documentation.



Test Assessment Ingredients

- Assessment methodology
 - Phase III – Report of findings and action plan
 - Actions and measures
 - Transition paths
 - Planning, timely, costs, and benefits

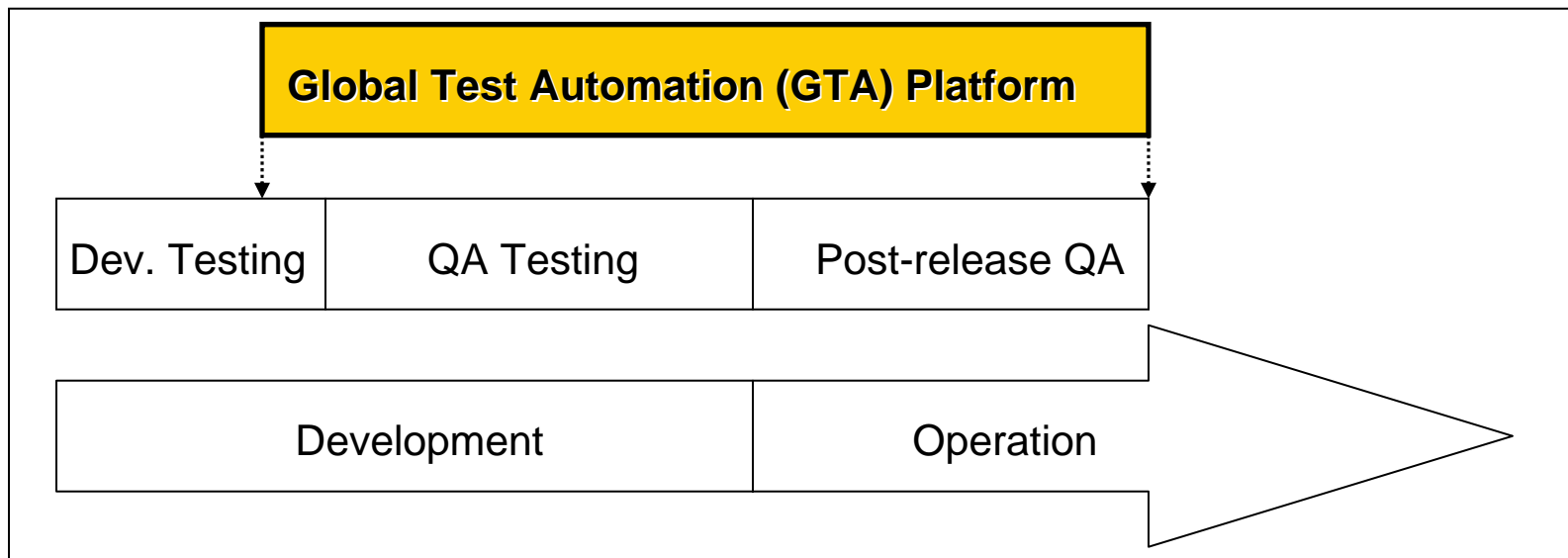
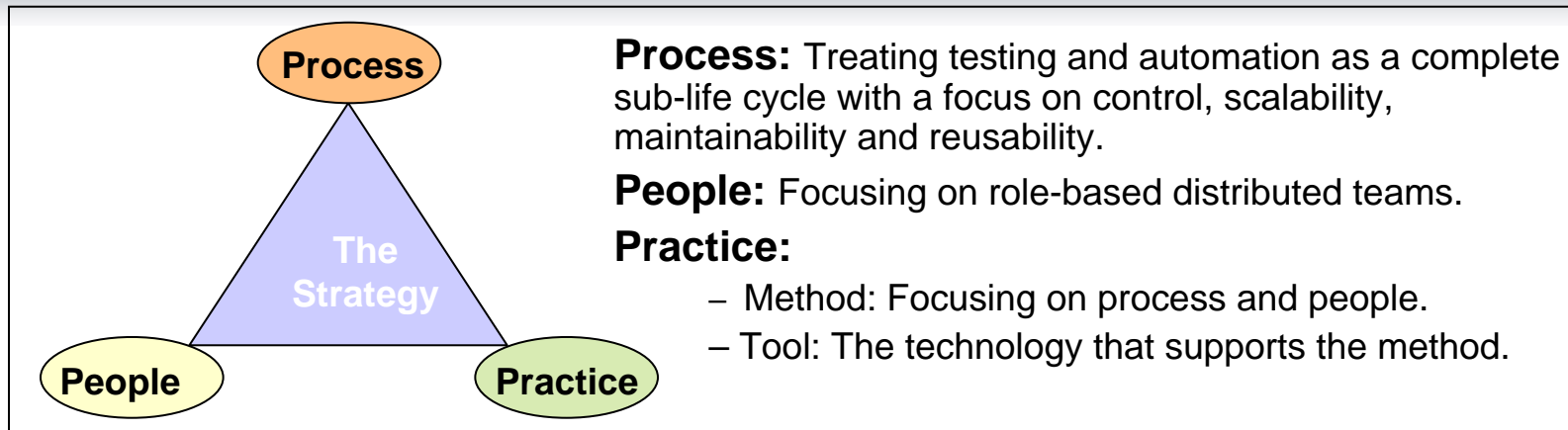


2. Test Process Alignment

Armed with data collected from the assessment, the next step is to look at the development and business process as a whole to help align the testing process.



The Process-Driven Test Strategy





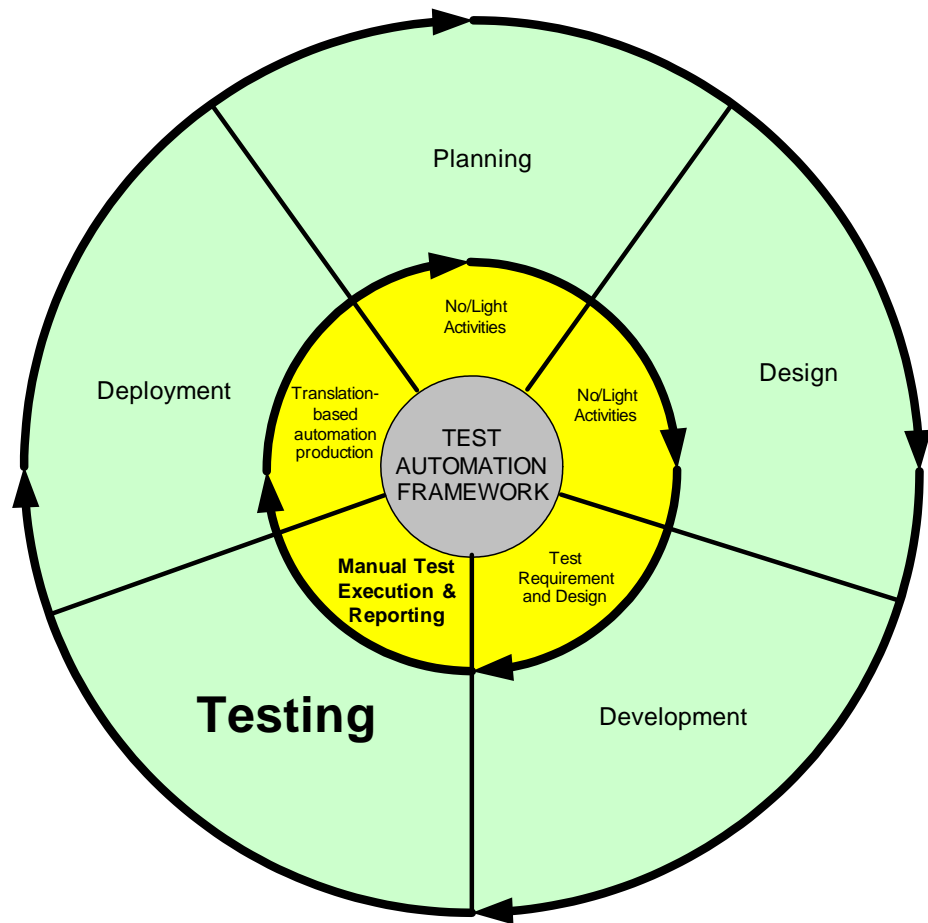
Process-Driven Test Strategy

- Process
 - Activities: Test Planning | Test Requirement | Test Design | Test Automation | Test Review | Test Execution—Manual and Automated | Test Report | Test Maintenance | Test Asset Handoff
- People
 - Roles: Distributed teams of business analysts, test engineers, automation engineers, leads, managers, and sustain-engineering staff
- Practice
 - Method: ABT or Action keyword-driven
 - Tool: Action-keyword enabling technology



Test Framework & The Application Lifecycle

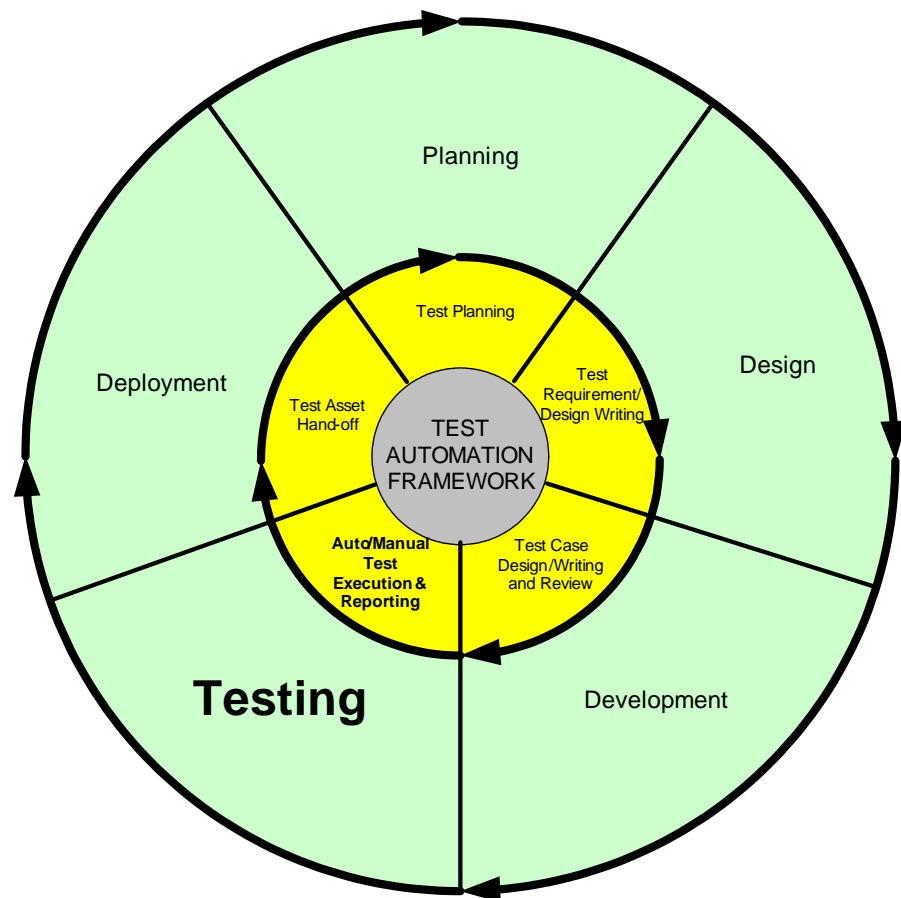
Integration of test and application development lifecycles without the Global Test Automation strategy.





Test Framework & The Application Lifecycle

Integration of test and application development lifecycles with the Global Test Automation strategy.





3. Leveraging Automation

Key Success Factors for Automation

- Accessible and maintainable test structure
- High degree of automation
- High degree of reuse
- At least as maintainable as the system under test
- Test specification separate from automation
 - Making sure to avoid double work between tester and engineer



Manual Testing Roles and Tasks

Roles

- Test engineers: Design/write and execute test cases; report bugs

Tasks

- *Test Case Production*
 - Design/Compose
 - Write/Transcribe
- *Test Execution*
 - Inputs (Run the test cases)
 - Setup environment
 - Generating inputs
 - Outputs
 - Problem identification or examine outputs
- Analysis
 - Problem analysis
 - Bug identification
 - Bug isolation
- *Bug reporting*



Automation Engineering Roles and Tasks

Roles

- Automated Test Engineers (ATE)—Creating (usually with a tool-specific programming language) automated tests or keywords (when keyword-driven framework is employed)
- Automation Framework Engineers (AFE)—Maintain the framework and support ATE's in the application of the framework.
- Automation System Support Engineering (low-level)
 - Support the integration of the framework with its native test automation agents or third-party test automation agents
 - Provide support of custom interfaces, objects, and/or platforms

Tasks—Automate Test Case Execution

- Coding approach without framework—Code or program the written tests to automate the test execution.
- Coding approach with framework (hybrid)
 - *Use a framework (such as keyword-driven)*
 - *Code or program the keyword*
 - *Code the written tests but leverage the pre-created keyword to boost automated test productivity and minimize automated test maintenance*
- Non-coding approach with keyword-driven framework—*Code or program and maintain libraries of keywords*



A Successful Automation Program

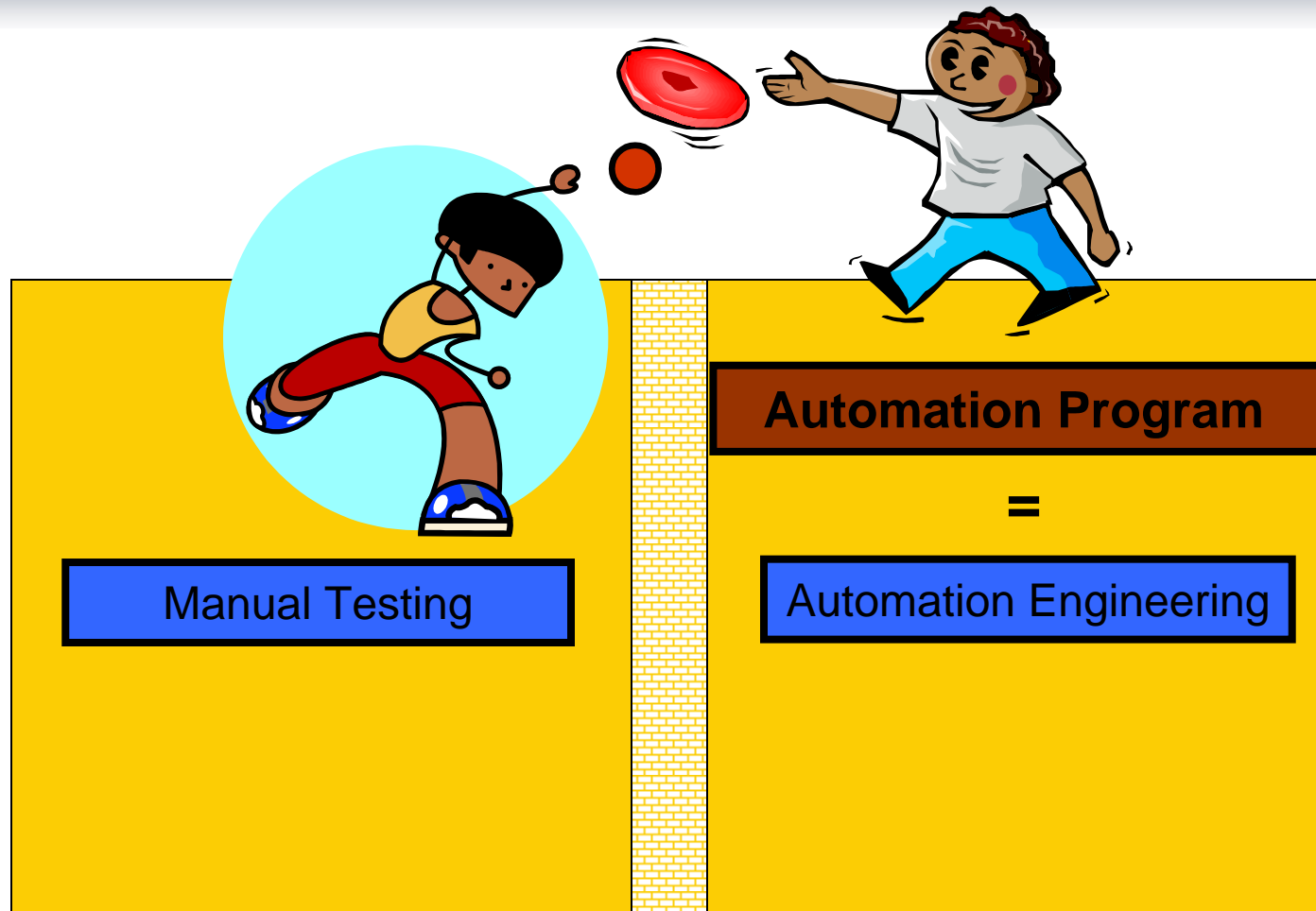
Objective

- *Achieve one or more of the following benefits: Reduce test cycle time, improve test coverage, and reduce overall testing costs*

Characteristics of a successful program

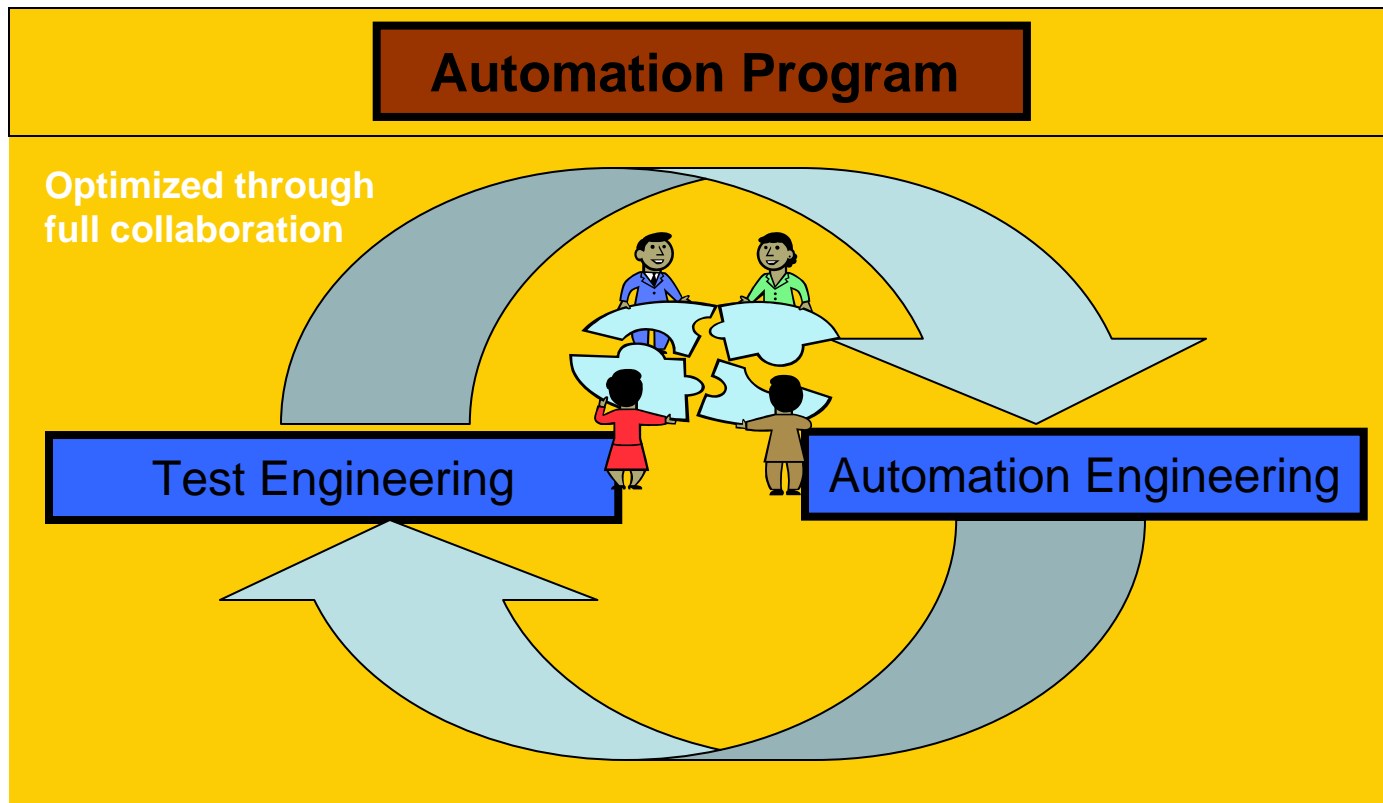
- *Fully utilize and leverage the talent and resource pool*
- *Maximize team adoption and collaboration*
- *Optimize all four areas of manual testing including test case production, execution, analysis and reporting*
- *Optimize control over the testing and test automation process*
- *Minimize scripting or programming*
- *Maximize reusability*
- *Minimize script maintenance*
- *Fully capitalize testware as an asset*
- *Fully scalable*

Example of a Partially Completed Program





A Complete and Scalable Program





4. Minimize Costs and Risks

- Build trust or use an offshore vendor you trust!
- Train the offshore staff or work with vendor who provides competent staff.
- Train all US leads who manage offshore resources as well as offshore leads.
- Address all of the communication issues including cultural and time zone differences, technical, English and project communication skills.
- Get a management tool or enabling framework that helps you facilitate project and task management.
- Get someone local who can facilitate.
- Get someone else to manage and be part of my team.
- Increase the productivity or savings from offshore team & reduce the headaches of my team in US.



5. Select the Right Toolset

You want to create the requirements for the tool based on your strategy of how your organization will optimally leverage

- test automation,
- the talent pool of test engineers and automation engineers,
- distributed teams such as offshore teams,
- and the process-driven strategy.

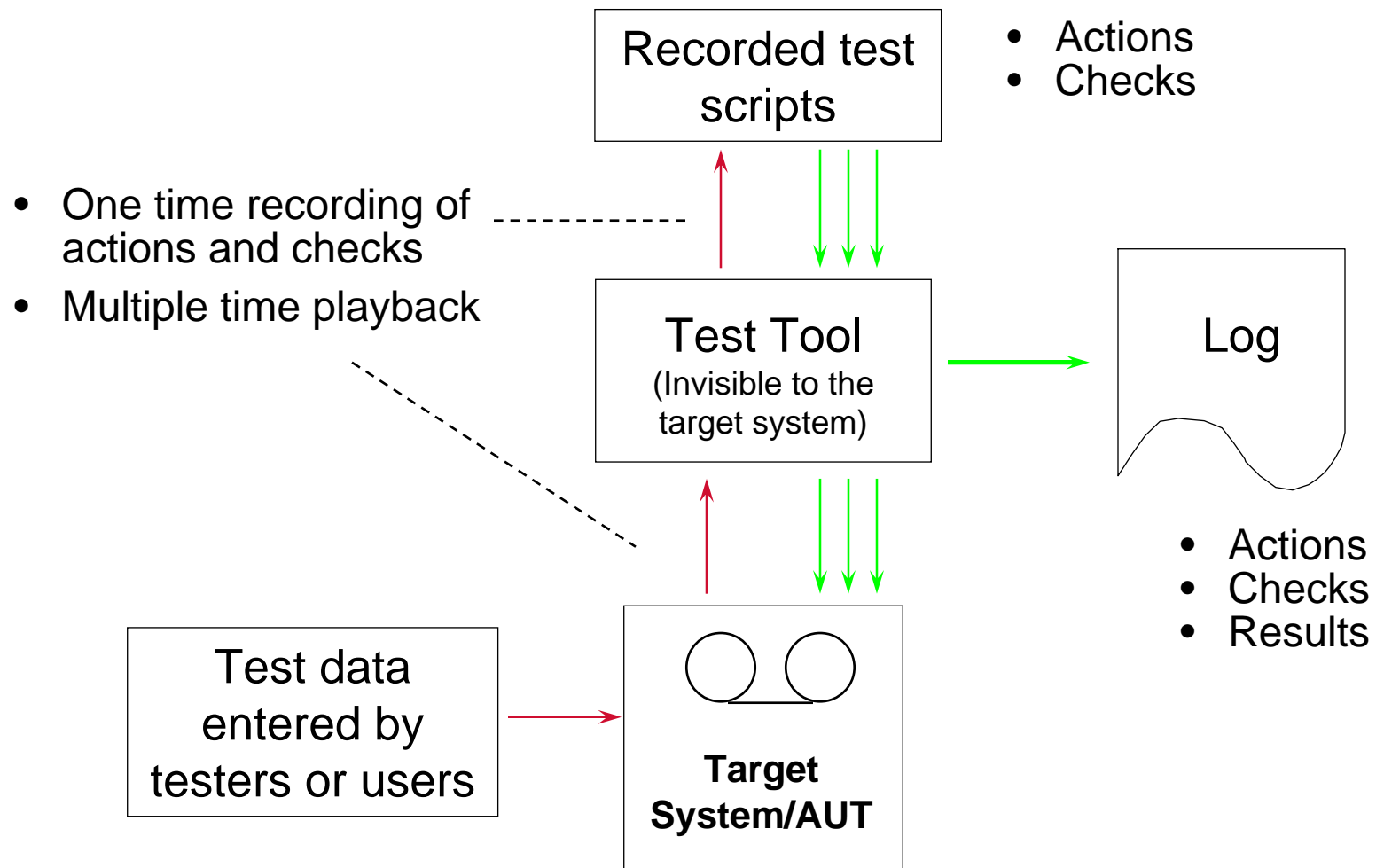


Tool Options

There are several different types of test automation tools available. They are identified below:

1. Action-Based Testing Approach – tools that utilize the hierarchical ABT techniques described in this chapter that focus on embedding the methodology for team-based test automation with great care on maintainability, scalability, reusability and visibility.
2. Record-Playback 2.0 Approach – tools that partially apply the keyword or similar approaches which focus on making it easy for non-technical engineers rather than maintainability, scalability, visibility and reusability.
3. Data-Driven Approach – tools that make use of captured data to execute the tests.
4. Scripted Approach – tools that use scripts to program and execute the tests.
5. Record-Playback Approach – tools that use a GUI to record mouse movements and keystrokes, and play them back automatically to run the tests.

Record/Playback




```
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"
```



R/P 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



Scripted Approach

- 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

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



Scripted Approach

- 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



(Scripted) Data-Driven Approach

- 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

Find [?] [X]

Find what:

☐ Match whole word only

☐ Match case

Find Next

Cancel

Find [?] [X]

Find what:

☐ Match whole word only

☐ Match case

Find Next

Cancel

Input Data

Find [?] [X]

Find what:

☐ Match whole word only

☐ Match case

Find Next

Cancel

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

```




(Scripted) Data-Driven Approach

- **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

- **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)



Record/Playback 2.0

- This is an extension of record/playback low-level commands are presented in a drop-list that can be easily accessed with you reading the code/script
- The low-level commands become keywords
- The objective is to make automation easier to use for non-technical staff



R/P 2.0 Pros and Cons

- Pros
 - A useful way to learn about the testing tool and automation
 - Easy adoption for non-technical staff
- 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

METHOD, NOT TOOL

AN INTRODUCTION TO ABT



Action Based Testing

- Separation of action, interface, 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



What is Action Based Testing?

- Action Based Testing (ABT) is an extended form of keyword-driven testing developed by LogiGear
- Keyword-driven test automation was conceived by Hans Buwalda and first introduced in Europe in the early 90's
- ABT is a result of many more years of research, practice, and technology honing of the original keyword concept, and lessons learned from the implementation of real-world projects



An Introduction to ABT

- The Action Based Testing methodology improves upon traditional test automation techniques by enabling non-automation experts to create automated tests, decreasing the amount of automation scripting required, and significantly reducing the amount of work necessary to update tests after a revision of the application under test.
- To understand Action Based Testing, it is beneficial to look at it in two ways:
 - First, as compared to software development.
 - Secondly, as compared to traditional test automation. By using analogy as well as comparison and contrast, it will become clear how ABT can produce superior results.

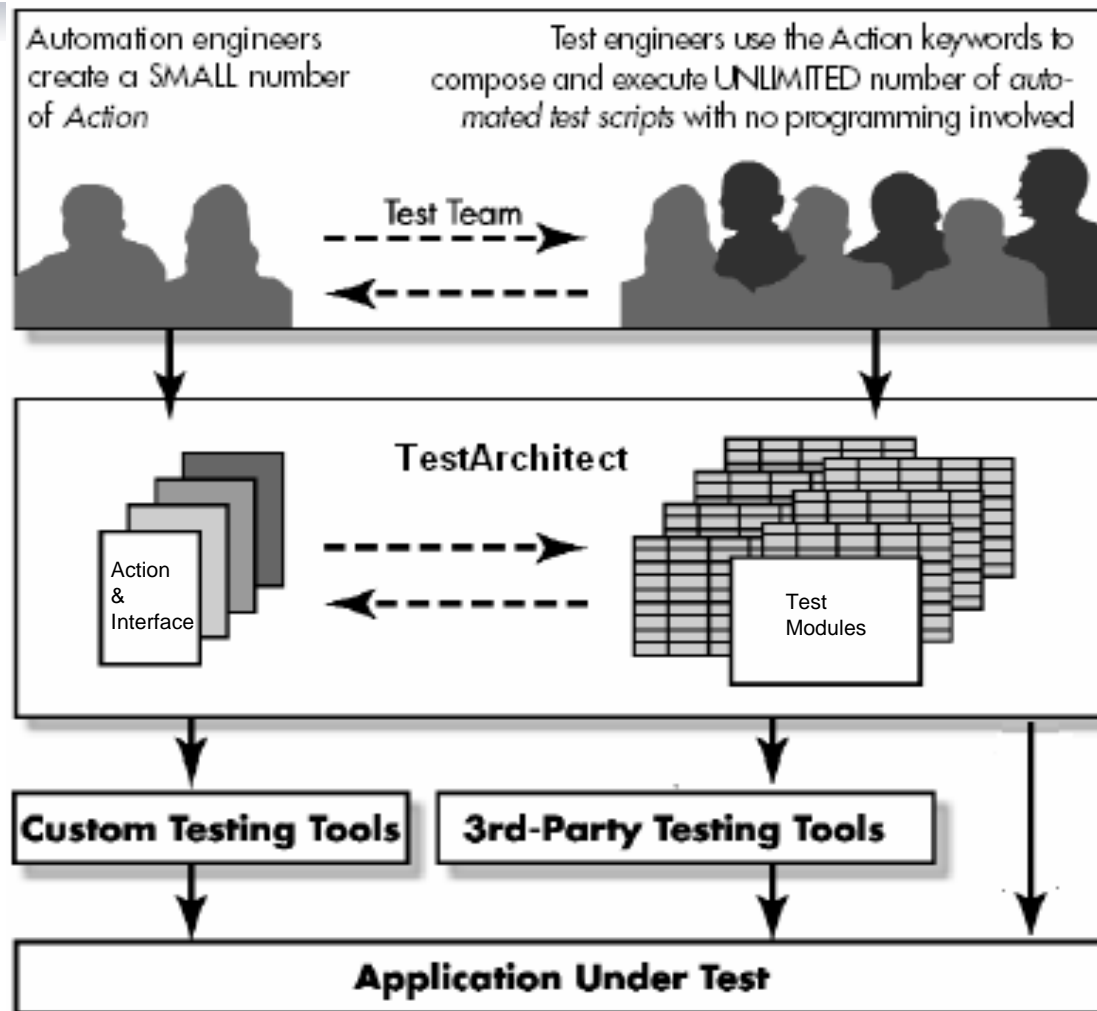


The Technology: TestArchitect™

Test Design and Test Automation—From TestArchitect's perspective, test designers and test automation engineers work within the following four environments.

- Test Module—Writing test cases
- Action Definition Worksheet—Creating action using a worksheet (without having to program or write code/script)
- Interface Definition Worksheet—Defining the interfaces of the AUT with which the actions interact
- Data Set Worksheet—Defining format and setting conditions of a data set to be used with data-driven test
- Harness (for scripting action using a programming language)

Division of Work



Software
testing
staff

TestArchitects™

Target system &
AUT



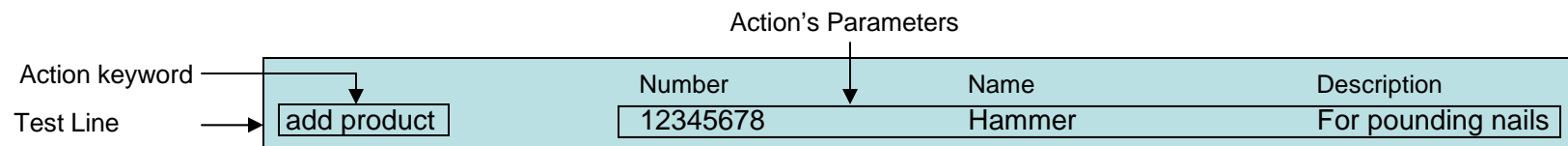
Division of Work

- Test engineers focus on test design
 - Work with Test Module and use actions to design test cases
 - Work with Data Set Definition
- Automation Engineers focus on automation
 - Work with Action Definition
 - Work with Interface Definition
 - Work with the harness



How Does ABT Work?

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



- **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

- Information about **Interfaces** such as Windows, Web or Java graphical user interface (GUI) or command-line interface (CLM) are later defined to enable actions to send inputs to and receive outputs from the target system and/or AUT



How Does ABT Work?

- **Test Case** information along with other test case related information is stored in a **Test Module**
- **Action** information is stored in an **Action Definition Worksheet** or in the **Harness** (more on this later)
- **Interface** information is stored in an **Interface Definition Worksheet**

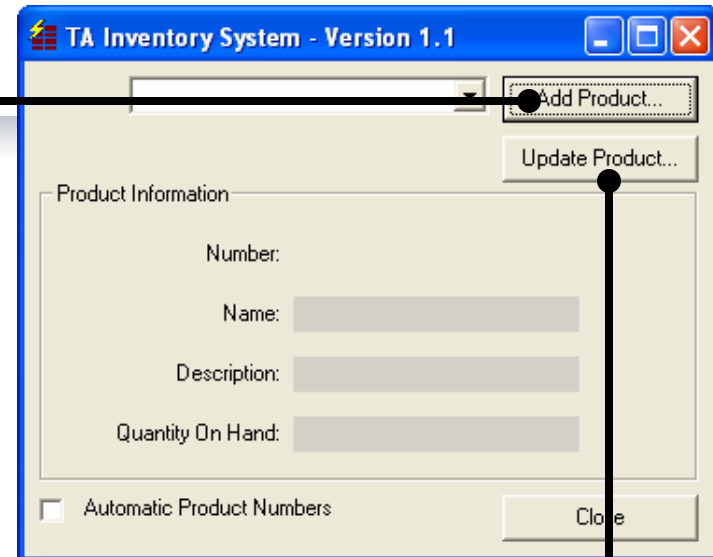
See an example —————>

ABT Example

■ An Application Under Test

This GUI application contains three dialog boxes:

- **Main** dialog box
- **Add A New Item** dialog box
- **Update Product Information** dialog box



TA Inventory System - Version 1.1

Buttons: Add Product..., Update Product..., Close

Product Information

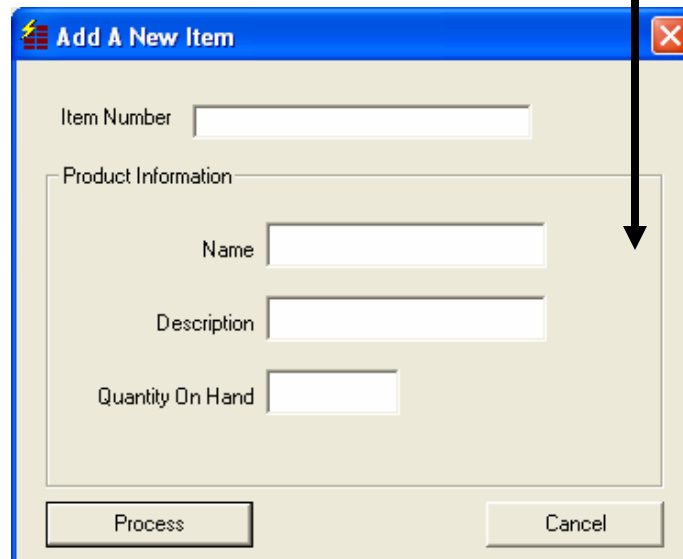
Number: _____

Name: _____

Description: _____

Quantity On Hand: _____

☐ Automatic Product Numbers



Add A New Item

Item Number: _____

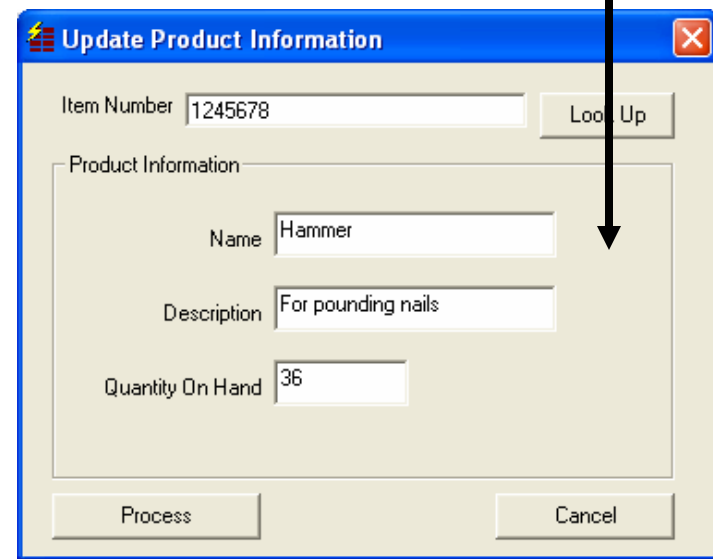
Product Information

Name: _____

Description: _____

Quantity On Hand: _____

Buttons: Process, Cancel



Update Product Information

Item Number: 1245678 Look Up

Product Information

Name: Hammer

Description: For pounding nails

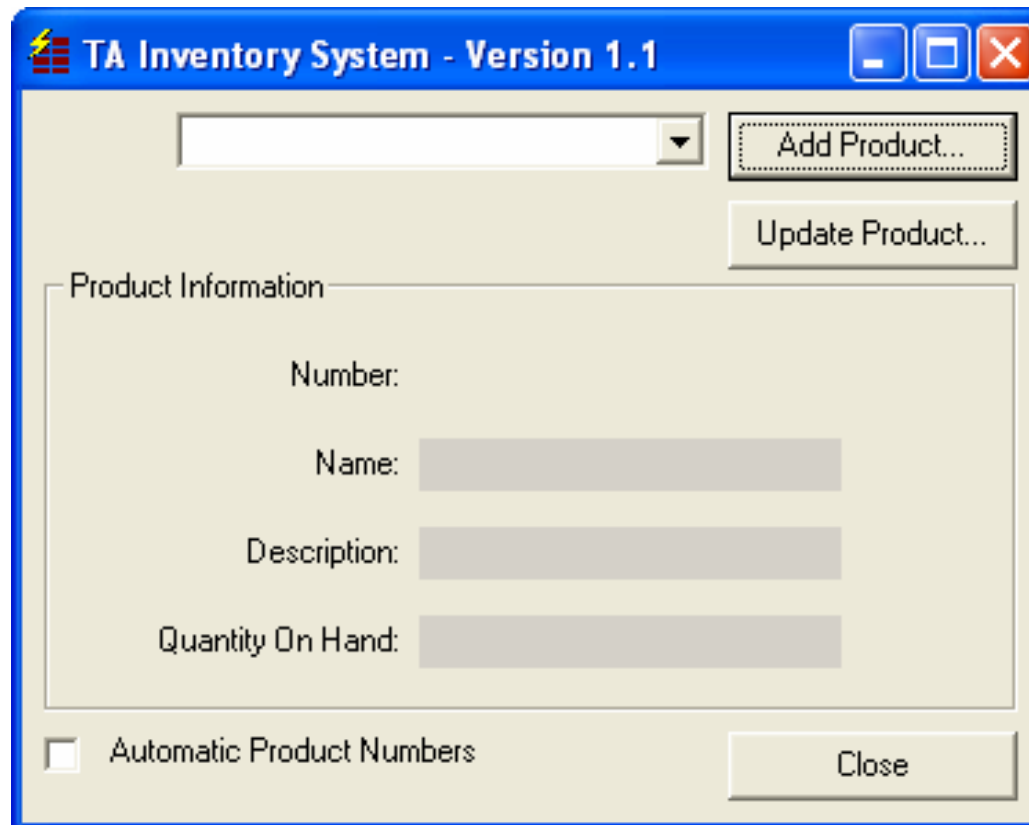
Quantity On Hand: 36

Buttons: Process, Cancel

Designing a Manual Test Case

1. Start Inventory

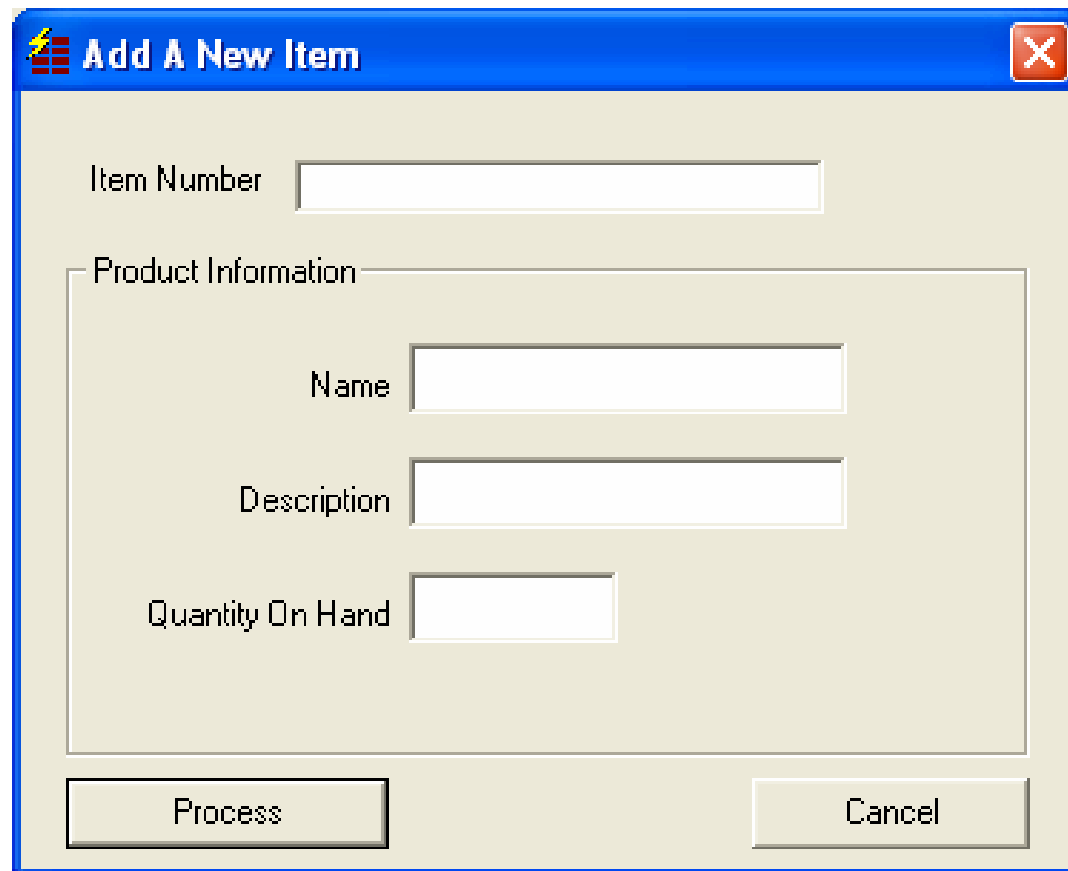
=>

A screenshot of a software window titled "TA Inventory System - Version 1.1". The window has a blue title bar with standard Windows window controls (minimize, maximize, close). Inside the window, there is a dropdown menu at the top left. To its right are two buttons: "Add Product..." and "Update Product...". Below these is a section titled "Product Information" which contains four labels and input fields: "Number:" (no input field visible), "Name:" (text input field), "Description:" (text input field), and "Quantity On Hand:" (text input field). At the bottom left, there is a checkbox labeled "Automatic Product Numbers". At the bottom right, there is a "Close" button.

Designing a Manual Test Case

2. Click the Add Product... button

=>

A screenshot of a software dialog box titled "Add A New Item". The dialog box has a blue title bar with a close button (X) in the top right corner. Inside the dialog, there is a form with the following fields: "Item Number" (a single-line text box), "Product Information" (a section header), "Name" (a single-line text box), "Description" (a single-line text box), and "Quantity On Hand" (a single-line text box). At the bottom of the dialog, there are two buttons: "Process" and "Cancel".

Add A New Item

Item Number

Product Information

Name

Description

Quantity On Hand



Designing a Manual Test Case

3. Enter

“87654321” in the Item Number text input field

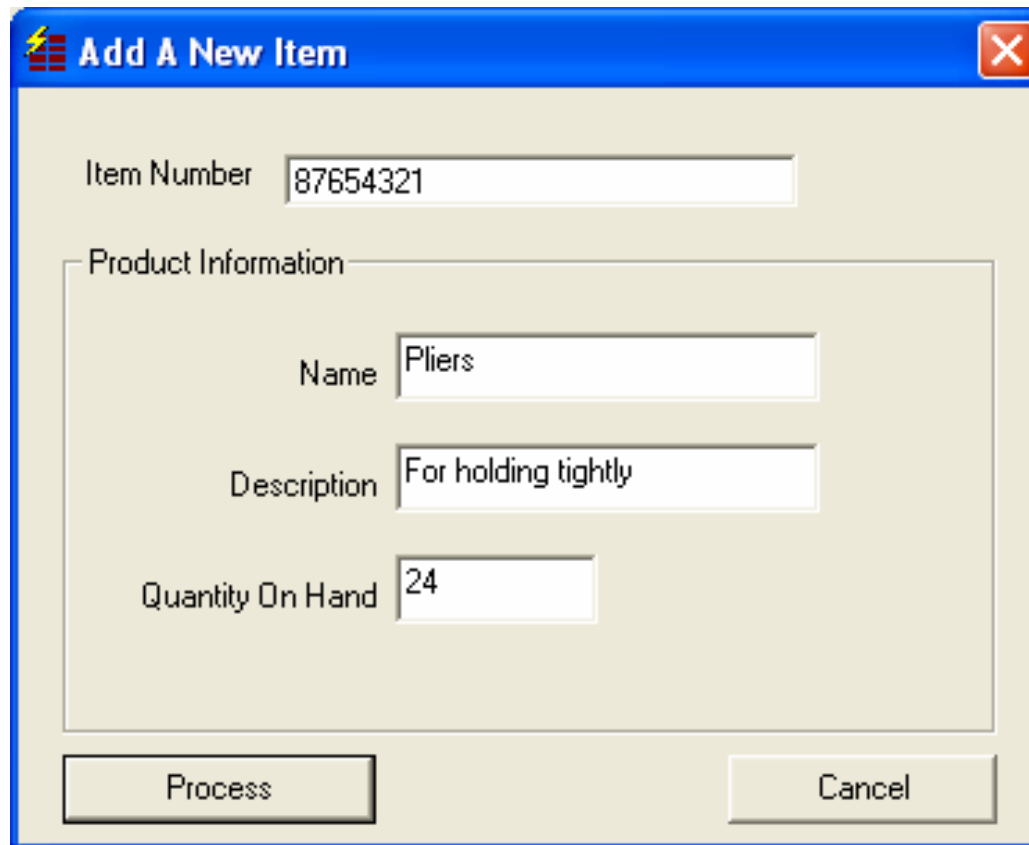
“Pliers” in the Name text input field

“For holding tightly” in the Description text input field

“24” in the Quantity On Hand text input field

Designing a Manual Test Case

=>

A screenshot of a software dialog box titled "Add A New Item". The dialog box has a blue title bar with a close button (X) in the top right corner. Inside the dialog, there are several input fields: "Item Number" with the value "87654321", "Name" with the value "Pliers", "Description" with the value "For holding tightly", and "Quantity On Hand" with the value "24". The fields are grouped under the label "Product Information". At the bottom of the dialog, there are two buttons: "Process" and "Cancel".

Add A New Item

Item Number

Product Information

Name

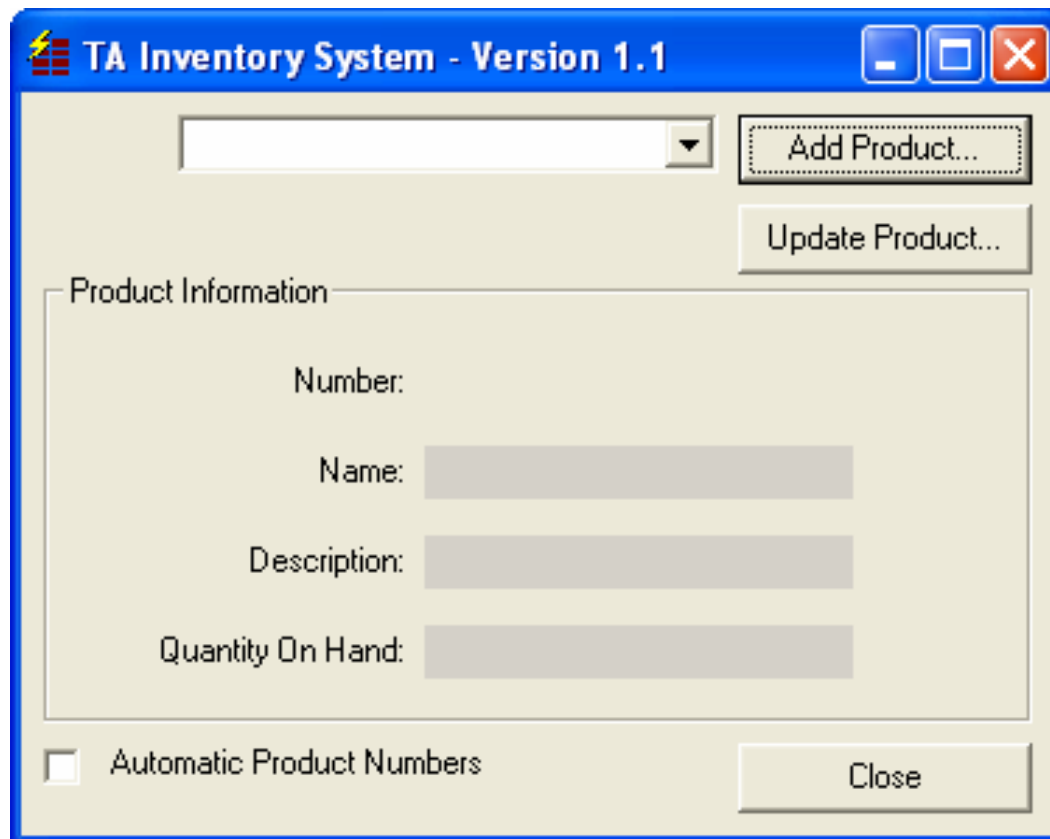
Description

Quantity On Hand

Designing a Manual Test Case

4. Click the Process button

=>



TA Inventory System - Version 1.1

Add Product...

Update Product...

Product Information

Number:

Name:

Description:

Quantity On Hand:

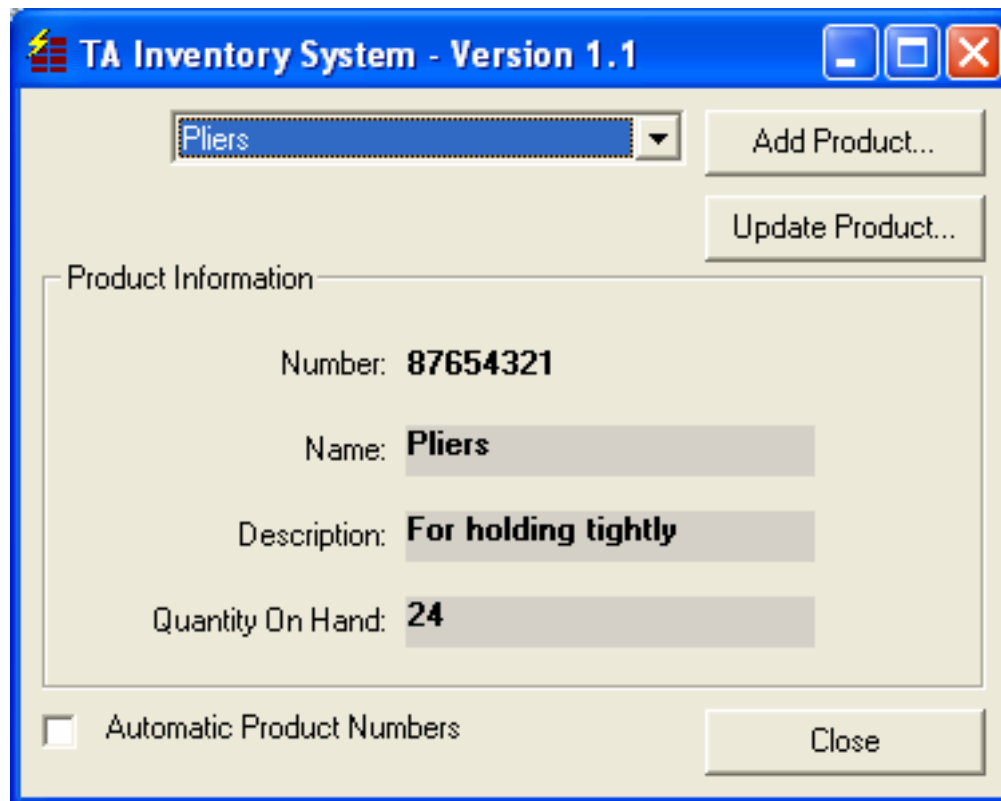
☐ Automatic Product Numbers

Close

Designing a Manual Test Case

5. Select the “Pliers” from the drop-list

=>



TA Inventory System - Version 1.1

Pliers

Add Product...

Update Product...

Product Information

Number: 87654321

Name: Pliers

Description: For holding tightly

Quantity On Hand: 24

☐ Automatic Product Numbers

Close



Designing a Manual Test Case

6. Check the accuracy of the newly created record
7. Exit the application by clicking the Close button



Designing a Manual Test Case

1. **Start** Inventory application
2. **Click** the Add Product... button
3. **Enter**
 - "87654321" in the Item Number text input field
 - "Pliers" in the Name text input field
 - "For holding tightly" in the Description text input field
 - "24" in the Quantity On Hand text input field
4. **Click** the Process button
5. **Select** the "Hammer" from the drop-list
6. **Check** the accuracy of the newly created record
7. **Exit** the application by clicking the Close button



Designing a Manual Test Case

First, it looks like we will use 6 actions in 7 test lines (or test lines) for this test case:

1. **Start**
2. **Click**
3. **Enter**
4. **Click** (Note: Same as in step 2)
5. **Select**
6. **Check**
7. **Exit**



Designing a Manual Test Case

- Secondly, we can combine steps 2, 3 and 4 into 1 step or 1 test line using a higher-level action that utilizes 2 low-level actions, Click and Enter. Let's name this new higher-level action "Add Product" which will add a new product record.
- Similarly, we can also combine step 5 and 6 into 1 step or 1 action line using one higher-level action utilizing 2 low-level actions, Select and Check. Let's name this new higher-level action "Check Product" which will check for the accuracy of the newly added product record.
- The new test case will look something like this:
 1. **Start**
 2. **Add Product**
 3. **Check Product**
 4. **Exit**



Designing a Manual Test Case

From a manual test case perspective, this is analogous to rewriting it as:

1. **Start** Inventory application
2. **Add Product** record button
Item Number: "87654321"
Name: "Pliers" in the Name text input field
Description: "For holding tightly"
Quantity On Hand: "24"
3. **Check Product** record for the accuracy of the newly created record
4. **Exit** the application



What Does Action Do?

Generally speaking, a typical action does one of the following:

1. Setting context such as initialization, configuring the toolset and/or the test environment
2. Sending to the application-under-test (AUT) inputs such as a mouse click event or data to an input field
3. Capturing and checking output information from the AUT



Test Cases w/ Actions versus Action Definition

STEP 1

9	TEST CASE	INV 001	Enter products using manual numbering
10	test requirement	TR-001	Can products be entered in the database
11	test requirement	TR-002	Are quantities displayed correctly
12	test requirement	TR-003	Can product information be updated
13	test requirement	TR-004	Can product numbers be specified manually

15	start inventory				
16		number	name	description	quantity
17	add product	87654321	Pliers	For holding tightly	24
18		number	name	description	quantity
19	check product	87654321	Pliers	For holding tightly	24
20		number	name	description	quantity
21	update product	87654321	Needle Nose Pliers	For holding in small places	36

STEP 2

22		A	B	C	D
23	check produ				
24		1	ACTION DEFINITION	add product	
25	exit inventory	2			

3		name	default value	description
4	argument	number		
5	argument	name		
6	argument	description		
7	argument	quantity		

Interface

8		interface	mode	
9	use interface	Inventory	windows	
10		source	control	destination
11	click	main	add	add product

Nested
Actions

12		window		
13	enter values	add product		
14		source		
15	click	add product		

item number	name	description	quantity
#number	#name	#description	#quantity
control	destination		
process	main		



Actions versus Interface Definition

	A	B	C	D	E	F
1	ACTION DEFINITION		add product			
2						
3		name	default value	description		
4	argument	number				
5	argument	name				
6	argument	description				
7	argument	quantity				
8		interface	mode			
9	use interface	Inventory	windows			
10		source	control			
11	click	main	add			
12						
13		window	item number			
14	enter values	add product	#number			
15		source	control			
16	click	add product	process			

	A	B	
1	interface entity	Main	
2	interface entity setting	title	ABT Inventory - What's In Stock
3			
4		to name	caption
5	interface element	add	Add An Item
6	interface element	update	Update An Item
7	interface element	end	End
8			
9		to name	caption
10	interface element	auto	checkbox
11			
12		to name	caption
13	interface element	products	local pos
14			combobox 1
15		level	caption
16	interface frame	1	product information
17			Product Information
18		to name	caption
19	interface element	number	local pos
20	interface element	name	textbox 1
21	interface element	description	textbox 2
22	interface element	quantity	textbox 3
23			textbox 4
24	end interface frame		

STEP 3

Interface
Interface Entity
Interface Element



Data-Driven Test

Data Creation

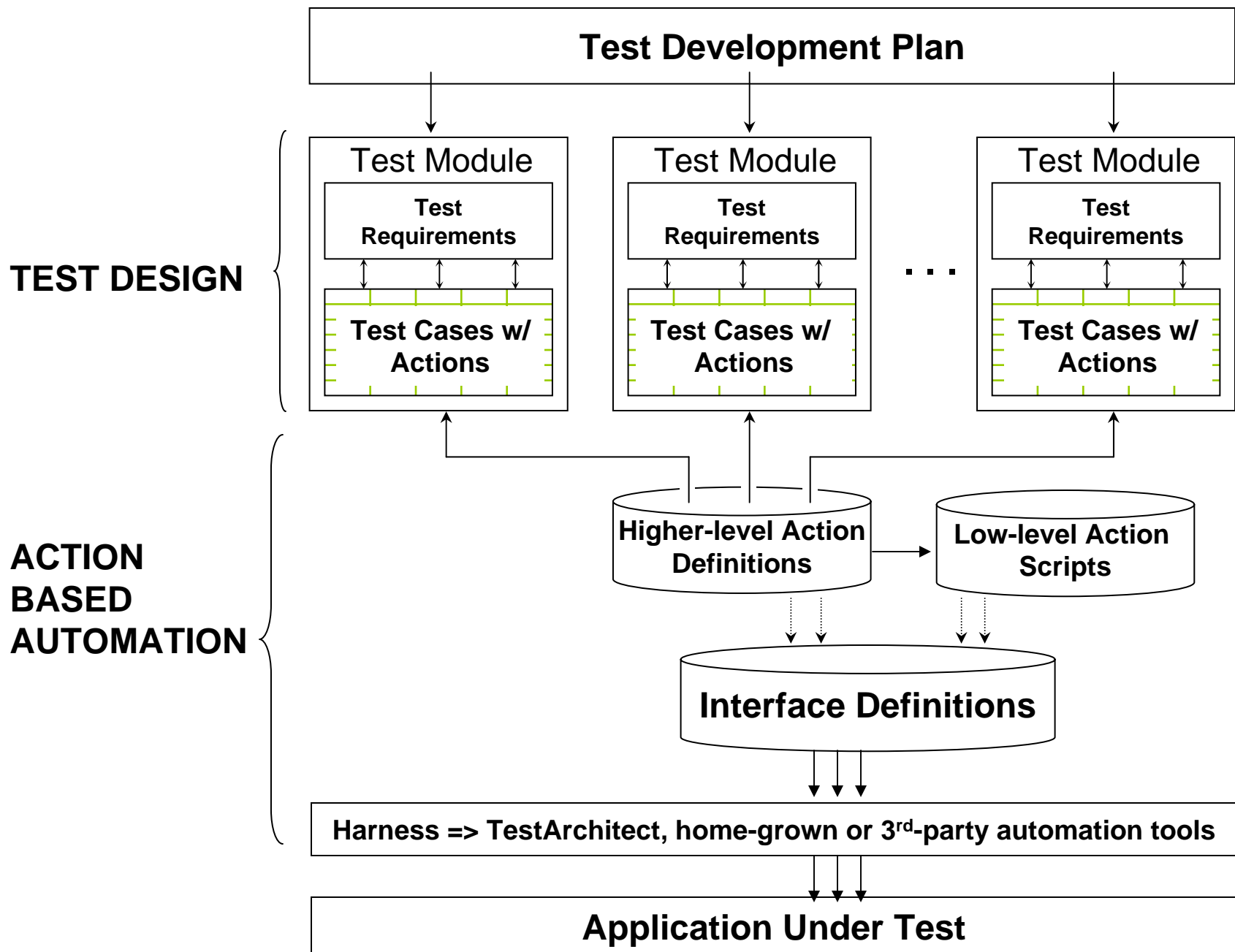
- A TestArchitect™ Data Set consists of data in tabular format. Each row is a record, and each column is a field within the record.
 - Each column within the data set has a specific name
- Enter data within your data set as shown in the screenshot below:

	A	B	C	D		
1	DATA SET	Inventory Data				(
2	VERSION					
3	STATUS	New				
4	LAST UPDATED BY	administrator				
5	LAST UPDATE DATE	7/8/2005				
6						
7		number	name	description	quantity	
8	row	57463292	Tires	Radial tires	8	
9	row	45892135	Axles	Between the wheels	10	
10	row	32198327	Wheels	17 inch wheels	24	
11	row	32014827	Taillights	Red plastic	326	
12	row	03512305	Headlights	For nighttime driving	22	

STEP 4
(If desired)

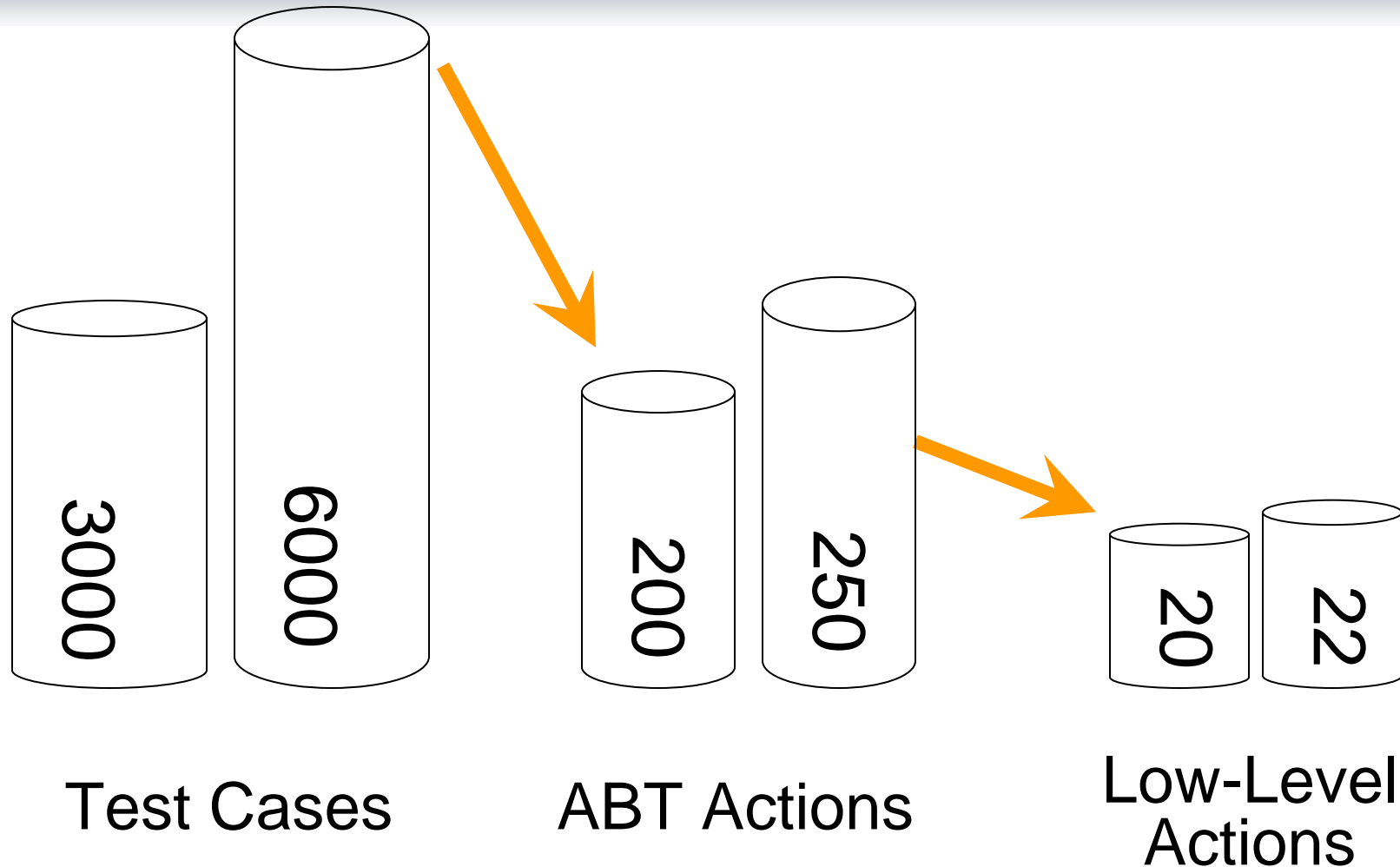
Data-Driven Test

TEST CASE		IDT 001	Data Driven Test Case			
use data set		name /Inventory Data	filter	high quantity		
start inventory						
add product		Number #number	Name #name	Description #description	Quantity #quantity	
check product		Name #name	Number #number	Description #description	Quantity #quantity	
exit inventory						
repeat for data set						
	1	DATA SET	Inventory Data			
	2	VERSION				
	3	STATUS	New			
	4	LAST UPDATED BY	administrator			
	5	LAST UPDATE DATE	7/8/2005			
	6					
	7		number	name	description	quantity
	8	row	57463292	Tires	Radial tires	8
	9	row	45892135	Axles	Between the wheels	10
	10	row	32198327	Wheels	17 inch wheels	24
	11	row	32014827	Taillights	Red plastic	326
	12	row	03512305	Headlights	For nighttime driving	22





Multi-level Action Approach





Case Study

Case Study: Centrifify Corporation

- US and offshore teams.
- Windows/Unix/Linux/Solaris 1.0 release.
- Custom automation platform built in 1 month.
- Global team designing tests within 3 months.
- 25 million tests run in less than 6 months.
- 80% of tests automated.
- Full test suite runs in 8 hours.
- Currently supports 35 operating platforms, automated tests run against every platform



Case Study

Case Study: Intuit, Inc.

"LogiGear's Action Based Testing (ABT) Solutions for test automation have had a significant positive impact at Intuit...as a result, we have a clearly defined, scalable and maintainable automation strategy to meet our current and future needs. Simply put, LogiGear's ABT Solutions has saved both time and money for Intuit. I wouldn't hesitate to recommend it to any of my professional colleagues, both within and outside of Intuit."

Heather Della Rocca, QA Manager, Intuit Inc., Turbo Tax Division



ABT Pros and Cons

- Pros
 - Highly scalable, best option for high-volume automation
 - Highly reusable, optimizing production throughput
 - Highly maintainable, reducing maintenance costs and time
 - Highly visible, providing control and manageability
 - Focuses on test design
- Cons
 - Clear life cycle and process needed
 - Not quite 100% code-free
 - Requires an understanding of Action Based Testing approach
 - Best ROI when employed with a large team and high volume test case



The Philosophy Behind ABT

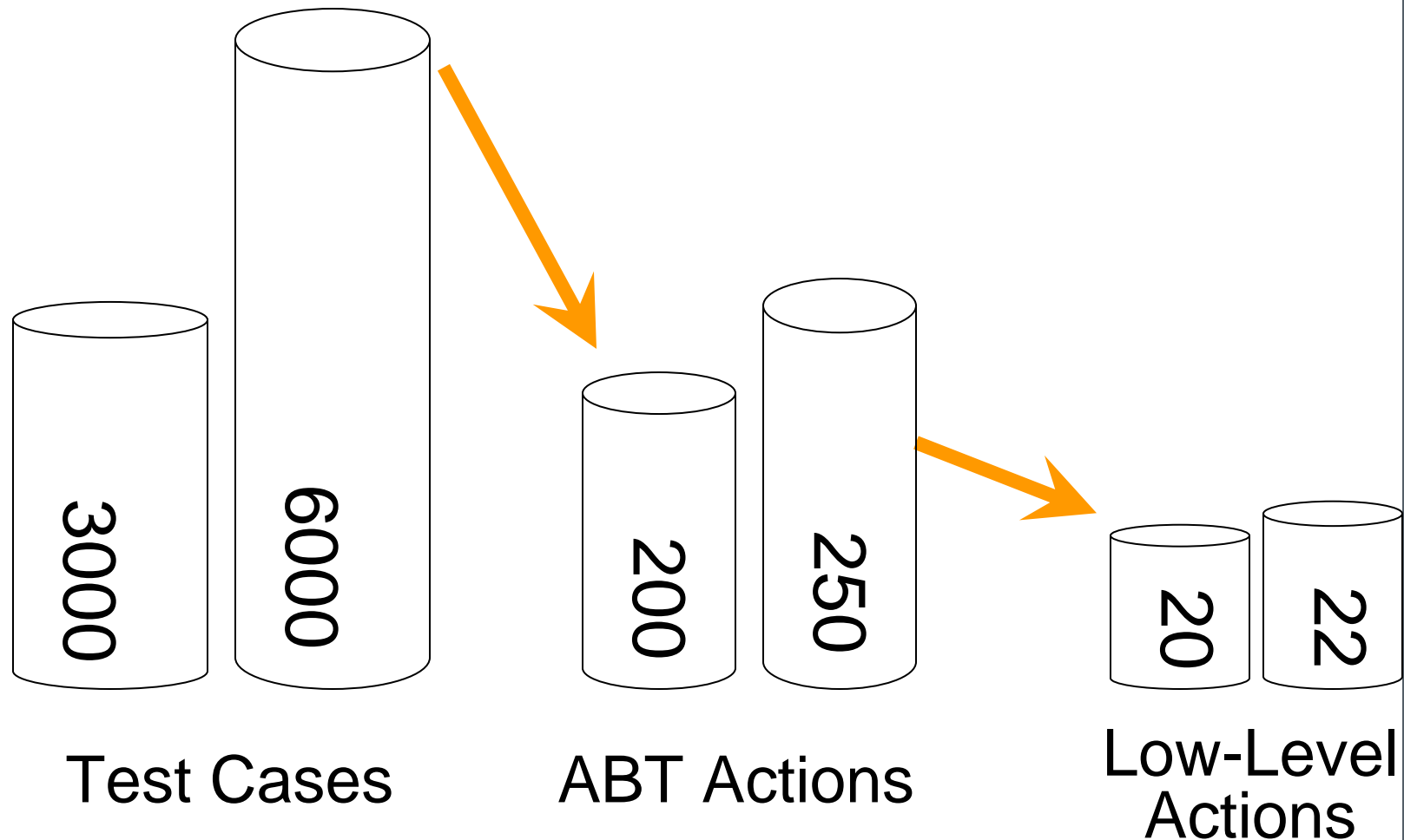
1. The Philosophy: *Tests Build-to-Last*
2. The Methodology: *Action Based Testing*
3. The Technology: *TestArchitect™*
 - *Action-keyword driven*
 - *Action-definition driven*
 - *Multi-level action driven*
 - *Data-driven*
4. The Application: *Global Test Automation*



Tests Build-to-Last

1. Tests are treated as product asset, along with the source code.
2. Tests, good or bad are dependent of the design.
3. Tests, manual or automated must be optimized for visibility, reusability, scalability and maintainability.
4. Tests must be automation-ready.
5. Tests, if they are worth automating, should follow the 5% rule:
 - No more than 5% of all tests should be executed manually
 - No more than 5% of all efforts around testing should involve automating the tests
 - No more than 5% of coded test scripts against non-coded test scripts.

Scalability Illustration

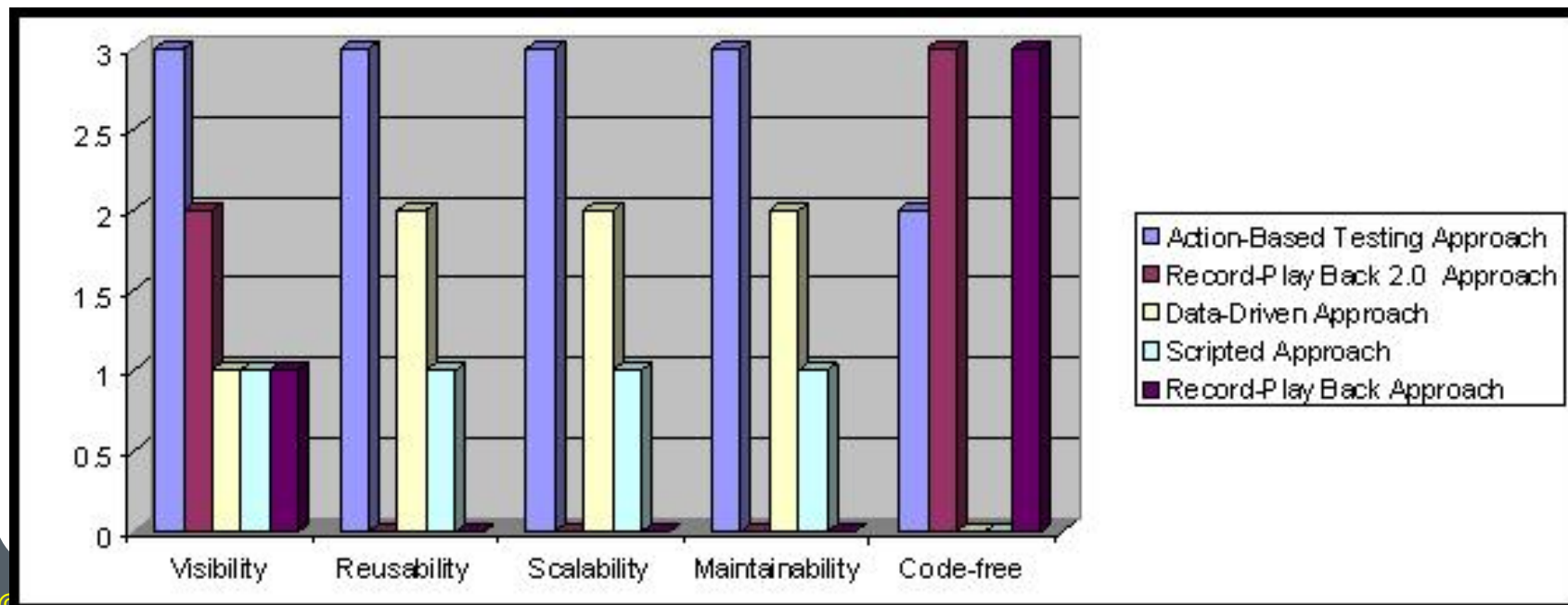


Tool Options

	Visibility	Reusability	Scalability	Maintainability	Code-free
Action-Based Testing Approach	3	3	3	3	2
Record-Play Back 2.0 Approach	2	0	0	0	3
Data-Driven Approach	1	2	2	2	0
Scripted Approach	1	1	1	1	0
Record-Play Back Approach	1	0	0	0	3

Legend

0: Very Poor
1: Marginal
2: Good
3: Very Good





Questions to Ask

- Does the tool support role-based test design, automation, execution, and management?
- Does the tool support traceability throughout the testing and test automation life cycle?
- Does the tool provide tracking and reporting mechanism for globally distributed testing teams' activities?
- Does the tool support seamless test asset transferring?
- Does the tool provide complete test automation development and QA testing life-cycle management capability?



Questions to Ask

- Does the tool support a complete integration testing and test automation while leverage the talent pool to focus on separate activities such as test design versus automation engineering?
- Does the tool leverage an embedded test and automation methodology?
- Does the tool help maximizing test and automation productivity?
- Does the tool help minimizing the maintenance effort?
- Does the tool provide accessible, visible and maintainable test structure?



Questions to Ask

- Does the tool support scalable production or a high degree of automation (the 5% rule)?
- Does the tool support a high degree of reusability?
- Does the tool provide comprehensive built-in playback support for many common application platforms such as Web, Window, .NET, UNIX, Linux, etc.?
- How flexible is the tool to be extended for uncommon platforms and/or interfaces such as custom software and embedded software (non-UI)?



Case Study

- **Case Study: OpenWave**
 - US and Offshore teams.
 - Windows/Unix/Solaris cross-platform
 - Reduced test creation time by 67%
 - Test execution time is now only 7 hours
(compare to 36 hours for the previous scripts)
 - Testing coverage is doubled
 - Expanded test suite runs 500% faster
 - Overall testing and test automation costs reduced by 31%



6. Secure/Develop Competency

- Your global staff must be equipped with adequate competency to do the assigned work.
- Your US staff must be equipped with adequate competency to collaborate and manage the offshore team.
- The global teams must seamlessly integrate.



6. Secure/Develop Competency

- Have a comprehensive training program in place for your teams.
- If you use an offshore vendor, make sure that their staff is fully trained, and a comprehensive training program is in place.



A Basic Curriculum Example

MODULES/TOPICS	Who should attend?
TESTING BASICS (Overview)	All
Why Test? What is the goal of testing?	
What is test coverage	
What is quality?	
What is a bug?	
Why are there bugs?	
Types of bugs	
Types of tests	
An overview of SDLC	
An overview of software testing organization	
How testing fits on SDLC	
Milestones	
Goal of testing in each phase	
Developer testing, tester testing, and test automation	



A Basic Curriculum Example

MODULES/TOPICS	Who should attend?
SOFTWARE TESTING SKILLS (Hard skills)	All
Test execution (running pre-written test cases)	
Bug identification	
Bug finding	
Bug analyzing and reproducing	
Bug reporting	
Bug reporting with TRACKGEAR	
Test case designing and writing	
Test case design with TestArchitect	
Status reporting	
Other reporting and communication	
Test documentation	



A Basic Curriculum Example

MODULES/TOPICS	Who should attend?
OUTSOURCING (Soft skills)	All
Understand the deliverables	
Understanding customer expectations	
The psychological issues of outsourcing	
Outsourcing models	
Roles and responsibilities	
Exceeding customer's expectation	
Delivering customer satisfaction	
Working in a distributed environment	
The LogiGear/LTRC difference	



Test Lead Training—Common Risks

COMMON RISKS (Quality & Productivity of the work)

Test team is not skilled enough to find bugs and come up with good test cases

Test team is not skilled enough to analyze anomalies.

Team does not write good bug reports.

Inadequate training and ramp up time for the team members



Test Lead Training—Top 10 Risk

US TEST LEAD

No or missing requirements & requirement/code creep.

Delays of the first testable builds and the software builds get worse over time, cut into testing time.

Poor testing and QA strategies.

Missed bugs due to poor coverage and/or quality of test cases.

Poor communication across functions, up and down the food chain.

Can't keep up with the work load due to lack of resource.

Personality conflicts among team members.

Team members get pulled into another project, side-tracked with production issues or hot fixes.

Forced to use outsourced resources but is not ready.

Automation program is ineffective: Time consuming but add little value, and maintainable and scalable.



Top 10 Risk to Manage

US TEST LEAD w/ OFFSHORE MANAGEMENT RESPONSIBILITY

Offshore work is not measurable or quantifiable so I have confidence that it's working.

Lack of visibility into day-to-day work.

Missing a competent lead/point-of-contact responsible to keep people on task and resolving on daily issues.

Lack of plans for downtime (power outage, internet, networks and/or servers go down, virus, build installation problems, blocking bugs, etc.)

Remote/offshore team loses access to onshore tools/resources (bug tracking, test cases, build server, test servers, etc.)

Offshore team did not tell me the truth--surprise!

Attritions or team members are distracted or reassigned to another project.

Personality, communication and culture conflict/clash between onshore and offshore teams.

Offshore team did not listen vs. Onshore team did not listen.

Language barrier (writing/speaking problems) get in the way of the work or productivity.



Test Lead Training—Top 10 Risk

REMOTE/OFFSHORE TEST LEAD

Information loss in the communication up and down the food chain (e.g., I did not hear about all changes, issues, problems, delays just like I was at local office.)

Lack of knowledge (e.g., I know what the product is about, the users, the technology and how to test it.)

Onshore Lead did not get visibility into day-to-day work. Therefore, does not have confidence about the work of my team.

Lack of plans for downtime (power outage, internet, networks and/or servers go down, virus, build installation problems, blocking bugs, etc.)

Remote/offshore team loses access to onshore tools (bug tracking, test cases, builds)

Onshore Lead does not listen to me, respond to me, and get answers for my questions timely.

Attritions or team members are distracted or reassigned to another project.

Personality, communication and culture conflict/clash between onshore and offshore teams.

The offshore team member will not buy into test strategy or follow instructions well, and do what they want.

Language barrier (writing/speaking problems) get in the way of the work or productivity.



7. Measure, Analyze & Optimize

- To improve you first need to have data
- To collect data you need measurement
- To know which measurement to take, you need to know
 - what you want to improve
 - the relationship between the measurement and the improvement indicator
 - what affects your measurement
- To make improvement decisions, you need to do root-cause analysis on your collected metrics



7. Measure, Analyze & Optimize

- If you are not the 'key' decision maker, you need to know who is the stakeholder. Who would care?
- You need to establish what number would be used as a success indicator because you will need to compare your data against referenced data.
- Metrics must be actionable
- Often time, the place to start is collecting metrics to find out where you are, so you can establish tangible goals (after you qualify the metric validity).



7. Measure, Analyze & Optimize

- What do you want to optimize?
 - QA/Test resource capability
 - Testing throughput or productivity?
 - Testing service quality?
 - The effect of the team capability on the released software (Strategy)?
 - The breadth of the test coverage?
 - The depth of the test coverage?



7. Measure, Analyze & Optimize

- What do you want to optimize?
 - Production throughputs?
 - Requirements?
 - Architectural and/or application design?
 - Specification?
 - Code?
 - The production quality (deliverables)?
 - Requirements?
 - Specification?
 - Code?
 - The effect of production throughputs and quality (Strategy)?
 - Deliverable quality?
 - Timeliness?
 - Cost efficient?



7. Measure, Analyze & Optimize

- What do you want to optimize?
 - Project management visibility and/or predictability?
 - Customer satisfaction?
 - Functionality?
 - Reliability?
 - Usability?
 - Performance?
 - Compatibility?
 - Security?



What Metric to Use and Why?

- What value do we want to associate with being “successful?”
E.g. miss a fewer number of bugs, or execute a larger number of tests
- Why *this* metric is relevant to the “success” value?
- What are we comparing against?
E.g., Wrote and executed 2,064 in 26 cycles. Is it a lot or two few comparing to what?
- What is value that we strive to meet?
- What about data integrity and dependency?



Some Examples

Project Management—getting the product out.
These are typically measures of software stability and activity

- How many test cases done
- Defect counts
- Hours tested against a build
- Code churn
- Requirements stability

Process Improvement

- Defect aging
- Requirement stability
- Valid defects found vs. test method
- Bugs by severity post release



In Summary

- You want to view your QA/testing and test automation activities as a complete sub-lifecycle.
- You want to develop a strategy that optimize your overall *team* throughputs in terms of quality of service, speed and cost.
- Use the seven-step process to develop your strategy: (1) Assess, (2) Align your test process, (3) Leverage automation, (4) Minimize costs and risks of global resources, (5) Select the right tools, (6) Secure/Develop competency, and (7) Measure, set goals and optimize.
- You want to institute a Global Test Automation strategy that focuses on process-driven test strategy, leveraging automation through Action Based Testing, the method, not the tool, selecting the right toolset , and securing and developing competency.
- You want to turn your strategy into a roadmap and commit to improvements.
- Changes will take time. So long that you don't get lost, you will reach your destination.



About LogiGear Corporation

LogiGear provides global solutions for software testing including the world-leading test automation solution with Action Based Testing. For over a decade, we've worked with hundreds of companies, from Fortune 500 to startups, delivering unique testing solutions that meet their unique needs. We double their test coverage, cut test time in half, improve quality and reduce cost.

LogiGear has built a reputation for offering the widest range of services in the software testing industry. Be it turn-key test automation, consulting, training, outsourced testing, or products, we partner with software organizations to create approaches that precisely meet their demands.

www.LOGIGEAR.com