Test Automation

Introduction

Test Automation Tools

Functional

- Rational software: Rational Quality Manager., Rational Functional Tester, Rational ClearQuest
- Quality Center, Quick Test Professional, Test Director
- QA RUN,QA Director
- Silk Test

Performance

- LoadRunner
- QA Load
- Application Performance Management
- Performance Studio, Rational Perf. Tester
- Silk Performer

SOA

- HP Service Test
- Rational SOA Test
- Parasoft Service
- iTKO Lisa

Security

- QAInspect
- WebInspect
- AppInspect

What is Automated Testing?

- Automated testing is the use of software tool to control the execution of tests, comparison of actual outcomes to expected outcomes.
- Automated testing typically involves automating a manual process already in place
- Automated Testing cannot replace manual testing as a whole, it adds value to it
- Automated testing may not be used for every scenario
- We @ University of Western Sydney use IBM Rational Functional Tester for Automation Testing

Why use Automated Testing?

- **Smoke Testing**
- **Speed up testing to accelerate releases**
  - Regression testing
  - Pre-production testing
- **Allows testing to happen more frequently**
  - Increased cost benefit
    - Scripts created once can be executed on different browsers (IE, Firefox, Opera etc)
    - Can be used to execute on different test ENV (SysTest, Integration Test, UAT etc)
  - Cheaper than manual testing in the long run
- **Improved Test Coverage in short amount of time**
  - Greater coverage
  - Manual Testing can concentrate on newly built functionality
  - Known execution time
  - More defects identified in testing phase

Which test cases to automate?

- Repetitive tests that run for multiple builds
- Tests that use multiple data values for the same actions (data driven tests)
- Identical tests that need to be executed using different browsers
- Tests that run on several different hardware or software platforms and configurations
- Tests that take a lot of effort and time by Manual Testing.

Which test cases not to automate?

- One-time testing
- Usability testing
- Ad hoc/random testing

Regression Automation

Automation Suite – When to go for it

- > 3 new builds/patches/fixes
- > 1 cycle
- Application Stable
- Run in Multiple hardware or software (including multi-Operating System or Multi-Browser)
- Application to meet SLAs?
- Repetitive tasks
- Test conditions re-used for IT/SIT/UAT?
- More Test execution efforts
- More of business flow testing and less of usability testing

Building Automation Suite

- Dependencies

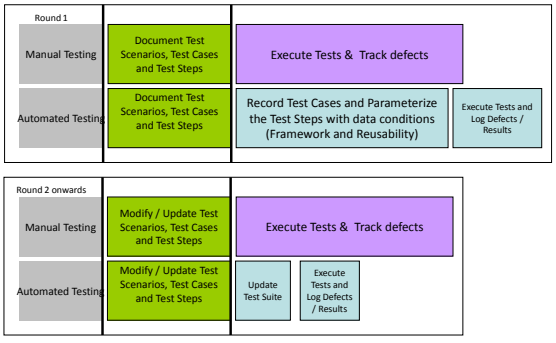
Note of Caution!

- Automation tools are NOT macro recorders, but fully functioning programming environments and must be treated as such.
- Record and playback ("point-and-click") will surely result in failure.
- Your automation engineer will need programming skills to:
  - create functions
  - access Win32 API functions,
  - read/write to files,
  - use ODBC / JDBC connection to make SQL calls,
  - utilize COM functionality
  - perform data correlation of complex SQL calls and web transactions, etc.
  - other programming techniques

There are 9 primary reasons for failure in automation.

1. Application instability
2. Application goes thru too many changes
3. Right automation tool is not chosen
4. Lack of structured automation methodology.
5. Test automation is not treated as a project with proper project planning (i.e. scope, resources, time-to market).
6. Testing is performed at the end of the development cycle (the waterfall method).
7. No modularization (use of functions) in automation scripts.
8. After initially creating automation suite, customer does not maintain the suite for future builds
9. Non availability of skilled automation testers

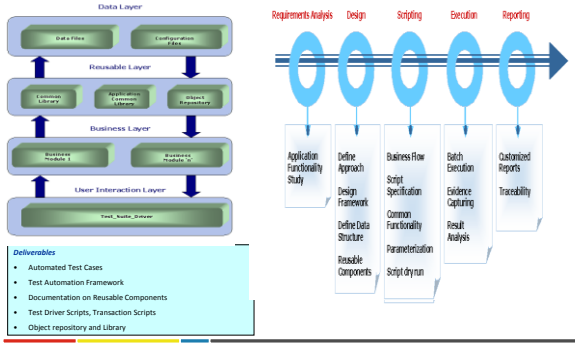
Manual vs. Automation Testing



Metrics for Automation

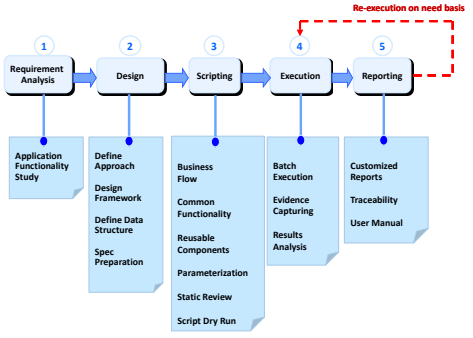
- Cycle Time Reduction
- Reduction in Test Execution Efforts
- Cost Savings
- > 98% Defect Unearthing Efficiency
- > 50% Productivity Increase

Automation Suite Development
– Design, Build and Execute (Not just record and replay)

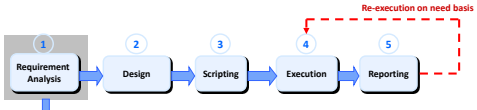


Deliverables
- Automated Test Cases
- Test Automation Framework
- Documentation on Reusable Components
- Test Driver Scripts, Transaction Scripts
- Object repository and Library

Regression Automation Life Cycle

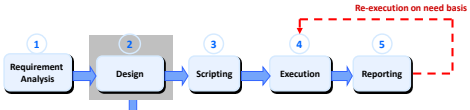


Regression Automation
– Requirement Analysis



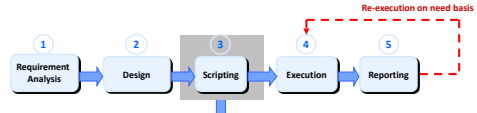
- Understand application under test
- Identify the testing tool to be used
- Identify scenarios/test cases to be automated
- Check automation feasibility of identified scenarios/test cases

Regression Automation - Design



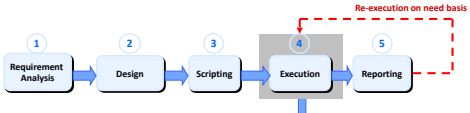
- Prepare automation test strategy
- Design automation framework (business functions & utility functions)
- Define data structure
- Identify the code conventions and standards for the testing tool
- Prepare program specification
- Prepare script verification checklist
- Prepare required test data

Regression Automation - Scripting



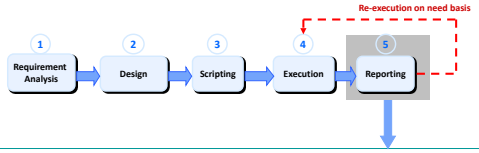
- Object Repository Preparation
- Generate common libraries – Sample Functions
- Scripting of Individual components
- Synchronize for inherent application delays
- Add verification points
- Plan recovery scenarios
- Integrate, review, run and validate the scripts

Regression Automation - Execution



- Run the scripts in uninterrupted mode
- Analyze the results
- Track defects to closure

Regression Automation - Reporting



- Generate customized reports on script execution
- Generate traceability between the scripts executed and scenarios identified
- Prepare and publish automation test report or release note
- Re-execute the scripts on need basis

Best Practices

- Do not Record, Write code
- Follow scripting guidelines and naming conventions
- Follow object repository guidelines and naming conventions to identify objects easily
- Provide traceability in result logs
- Data driven tests with the flexibility to test for various business conditions on the same transaction
- Modular approach in test scripts with plug and play feature for future enhancement
- Use checklists for code review and Integration testing of Test scripts

Benefits of Test Automation

Tangible benefits

- Speed and Accuracy
- Accessibility
- Accumulation
- Manageability
- Early Discovery of issues
- Repeatability
- Availability

Intangible benefits of test automation

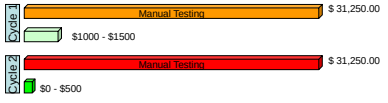
- Formal process
- Retention of customers/user trust
- Greater job satisfaction for Testers

How Automation saves Time & Money ?

Claim Risk Prediction Tool – ABC

- The fuzzy logic in the ABC predicts the Risk involved in a claim as HR, LR, FT or Catastrophic which is crucial for the business to do better business.
- Business decided to take a sample from 3 months (3000 records / combinations identified) to test the logic.
- Testing each record manually takes about 5 min on an average
- Manual testing of 3000 records X 5 mins = 31.25 days (@ 8hrs /day)

- Testing each record by executing Automated script takes about 2 min on an average
- Automation development time = 1 day
- Automation Testing 3000 records X 2 mins = 4.16 days (@24hrs /days)
  = 1 day (If run on 4 PCs etc.)



Performance Testing

Performance: Observed Trend

- More and more complex IT implementations
- Customers are more demanding
- Cost of Hardware & Software Infrastructure Maintenance is increasing
- Volumes are increasing
- Application complexity is increasing
- Existing monitoring framework ⇒ Limited to System Monitoring
- Buy rather than build
- Performance problem ⇒ Add hardware
- Performance: A reactive process rather than a proactive process

Workload Estimation

USERS:

- Types of Users
- Number of Users Per Type
- Number of Concurrent Users
- Frequency of Interaction
- Complexity of Interaction

VOLUMES:

- List of business entities
- Volume per entity
- Growth Rate
- Retention period

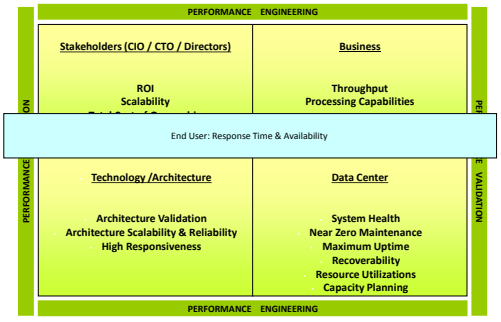
TRANSACTION & REPORTS:

- Types of Transactions
- Transaction Mix
- Transaction Rate – Avg, Peak, Growth
- No. of Interactions Per Transaction
- Transaction Complexity

BATCH OPERATIONS:

- Number of batches
- Periodicity / Schedule of each batch
- Complexity of each batch
- Backups and their frequency

Performance Engineering – Parties Addressed



Protocols & Monitors

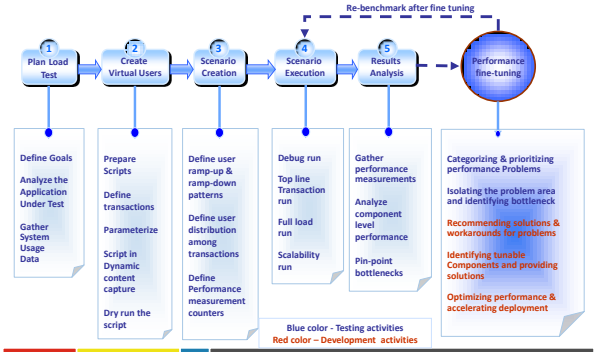
- Protocols –
  - ERP / CRM – SAP, Oracle, Seibel and People Soft
  - Web – http, https, xml, Citrix ICA, SOAP and WAP
  - Middleware – EJB, CORBA, COM, RMI, Tuxedo and MQ Series
  - Database – Oracle, MS-SQL, DB2, Informix and ODBC
- Monitors –
  - OS – Windows (NT, 200x, XP), Unix and Linux
  - Network – SNMP
  - Middleware – Tuxedo and EJB
  - Firewall – Checkpoint
  - Web Servers – MS-IIS, IPlanet, Apache and Oracle 9iAS
  - Database – Oracle, MS-SQL and DB2
  - Web Application Servers – Web Logic, Web Sphere and MS ASP

Performance Requirements Analysis

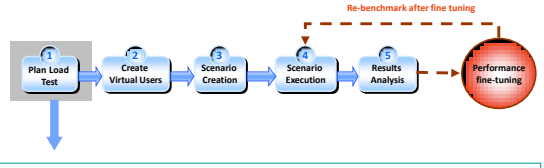
PERFORMANCE TARGETS

- Average Response Time per Interaction – Intranet and Internet
- Is the System Stable Enough (Acceptance)
- Does the New Version impacts Response time (Regression)
- Capacity Planning (At what Points Does the Performance degrades)
- Resource Consumption Targets (Configuration Sizing) :
  - Average CPU Utilization < X% across all CPUs
  - Average Memory Utilization < Y%
  - Total Heap Size should not be < Z%
  - Average Network Bandwidth Consumed < N Kbps

Performance Testing Process – Test, Execute & Tune Approach

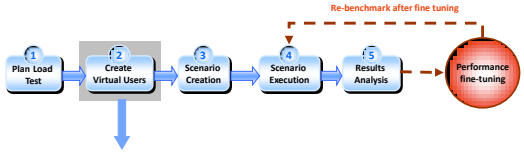


Performance Testing Processes – Plan Load Test



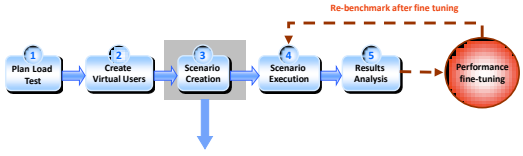
- Quantitative Definition of Goals
- Analyze the Application Under Test
- Gather System Usage Pattern for Emulating Real Time Load Pattern

Performance Testing Processes – Create Virtual Users



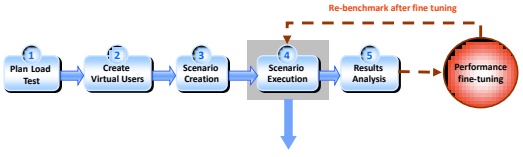
- Record the Scripts and Capture Real Time User Actions
- Define Transactions
- Parameterize Script
- Capture and Handle Dynamic Content
- Dry Run the script to Verify Successful Execution

Performance Testing Processes – Scenario Creation



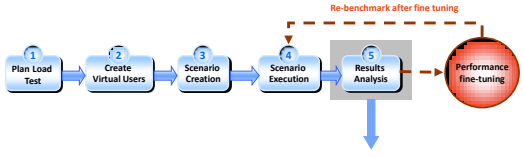
- Define User Ramp-up & Ramp-down Patterns
- Define User Distribution Among Transactions
- Transaction Rate
- Define Performance Measurement Counters

Performance Testing Processes – Scenario Execution



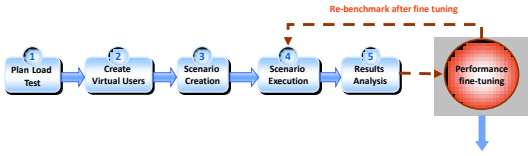
- Quantitative Definition of Goals
- Execute Debug Run
- Execute Top Time Transaction Run
- Execute Full Load Run
- Execute Scalability Run

Performance Testing Processes – Results Analysis



- Gather Performance Measurements of the Run
- Analyze Component Level Performance
- Pin-point Bottlenecks
- Identify Problem Areas

Performance Testing Processes – Fine Tuning



- Categorizing & Prioritizing Performance Problems
- Isolating the Problem area and Identifying Bottleneck
- Recommending Solutions & Workarounds for Problems
- Identifying Tunable Components and Providing Solutions
- Optimizing Performance & Accelerating Deployment

SOA Testing

What is SOA?

There is no widely-agreed upon definition of service-oriented architecture other than its literal translation that it is an architecture that relies on service-orientation as its fundamental design principle. Service- orientation describes an architecture that uses loosely coupled services to support the requirements of business processes and users. Resources on a network in an SOA environment are made available as independent services that can be accessed without knowledge of their underlying platform implementation.

Difference from conventional Testing

- **Infrastructure challenges** as well as use of XMLs / WSDLs as test inputs and outputs
- It is difficult to pinpoint a **'stress point'** due to multitude of services that make a workflow and this is worsened as individual services are added / changed the existing workflow
- Testing involves not only unit level testing, but also testing the business rule that **embodies** the service; thus testers need to have technical as well as business knowledge
- Services **need** to be **tested earlier** in the lifecycle because the cost of resolving defects in deployed services can be high

Solution

- Services to be tested in **isolation** for business rule in addition to unit testing and integration testing of the constituent components
- Since services are built for re-use and high volumes, they are tested for **performance**, **security** and **interoperability** along with functional testing
- In- House automation simplifies writing of test cases and validation of results resulting in **increased efficiency** and **accuracy**
- Testers are **trained** on technical as well as business knowledge so that they can navigate application innards, at the same time understand how well services embody business processes

Thank You