Software Test Automation- Best Practices

Leonard Fingerman
Email: lfingerman@gmail.com

Cost vs. Quality Dilemma

". . . and we can save 700 lira by not taking soil tests."

Why Automate Testing ?

- Improve software quality while reducing time to market …not simply increase test coverage

4. Improve reliability and consistency of testing processes

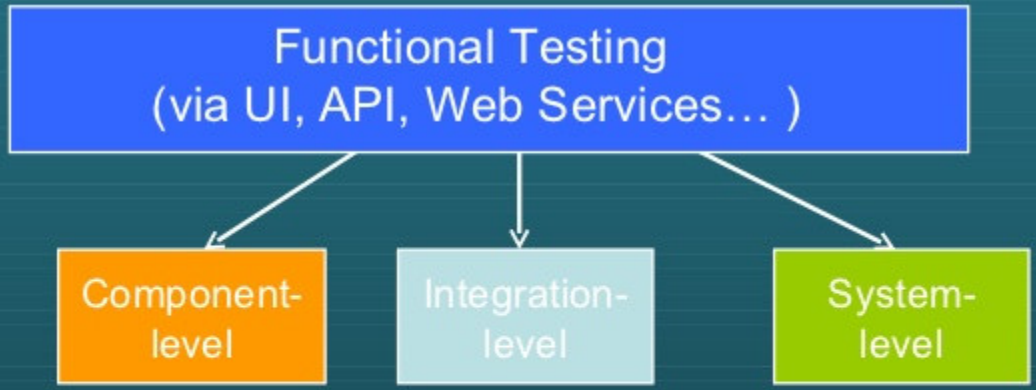
3. Allow manual testers do more complex & more suited for human kinds of testing (exploratory, usability etc.)

8. Enforce SLAs (service level agreements)

Bottom line is save your company $$$

Common Types of Automated Testing

- Unit Testing (xUnit, TestNG …)
- Performance Testing (LoadRunner, JMeter …)
- Web Service Testing (SOAPUI, HP ST, iTKO LISA…)
- Security Testing (WebInspect, WireShark…)

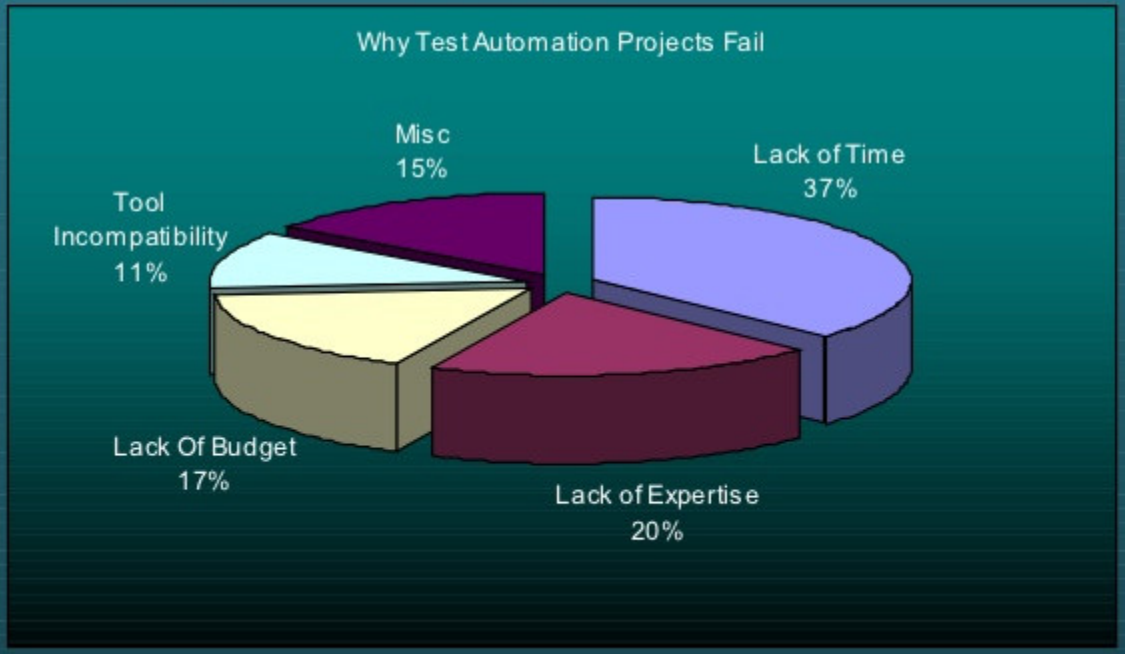


Common Functional Automated Test Tools

Tool	Pros	Cons
IBM/Rational Functional Tester (RFT)	•Built as Eclipse Plug-In with full IDE, Java support & Source Mgmt •Supports Web 2.0, Java or .NET applications •Full GUI Object Map repository	•Insufficient browser support •License cost
HP/Mercury Quick Test Pro (QTP)	•Supports Web 2.0, Java or .NET applications •Full GUI Object Map repository •Seamless integration with QualityCenter	•VisualBasic scripting is limited •No IDE (changing in release 10) •License cost
Selenium RC & IDE	•Good browser support •Good language support (Java, Ruby,C#) •Can be easily extended as JUnit suite •Open-source (no license cost)	•No GUI Object repository •Only web-based application support •Tool support is limited

Typical Test Automation Pitfalls

- You selected a Tool…now what?
- A tool is supposed to do it all, right… ?
- Don't fall into tool vendor's sales pitch …remember Capture & Playback is not real test automation
- According to recent IDT study (www.idtus.com)



Tips for Test Automation Strategy

* Essentially automated test system is a piece of software designed to test your application → treat it as such

– **Evaluate** an automated tool
  - POC (proof of concept) based on your native apps
  - Choose a tool that fits your business requirements & internal technical expertise

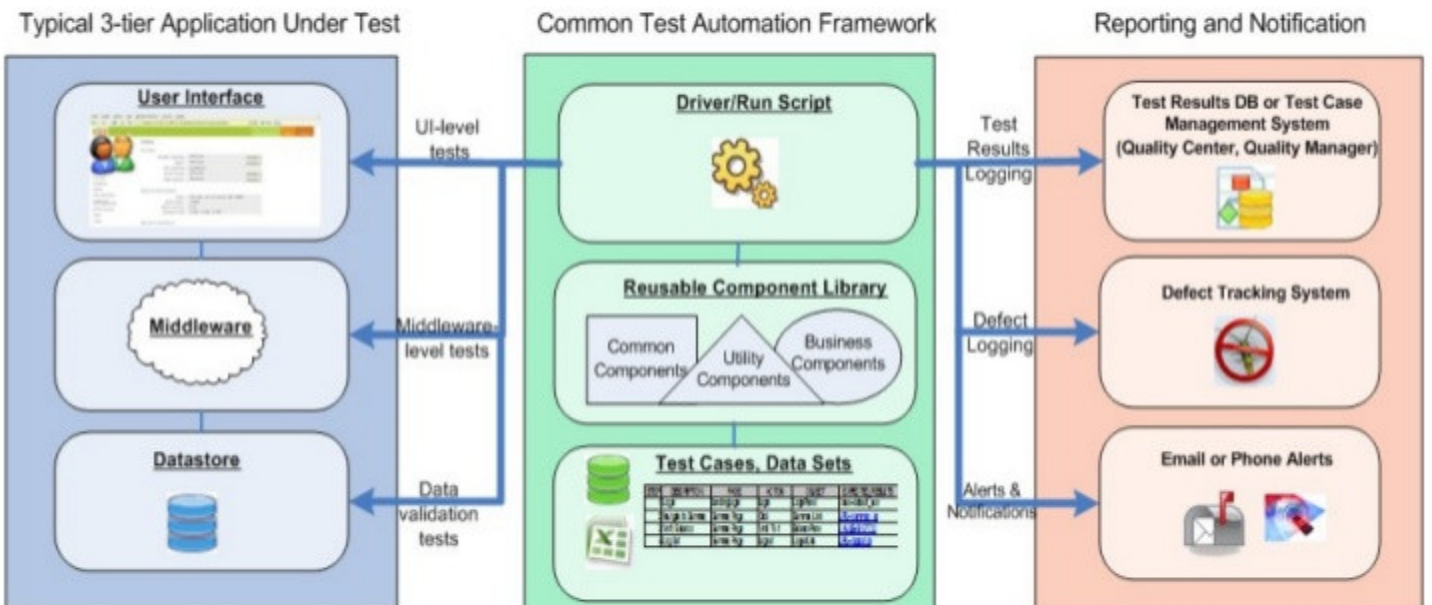
2. **Plan**
  - Establish goals & objectives; budget; use or hire competent resources
  - Be ready to have quality metrics to validate the ROI (return of investment)

3. **Design**
  - Keep in mind the user community of TA system (Usability)
  - Choose an architecture that reduces maintenance & extensible

– **Implement**
  - Practice OOD principles of reusability & modularity (employ encapsulation & abstraction)
  - Introduce version control of your code
  - Establish code review process

– **Test & Maintain**
  - Unit test your code
  - Constantly re-evaluate reasons of maintenance & improve TA architecture to minimize maintenance. Application testability usually plays a big factor.
  - Evaluate defect rate with root cause analysis

High-level Diagram of Test Automation Architecture



2nd Generation Automation Frameworks "Data-driven"

- Automation is data-centric
- Test Tool – dependent (due to flow control in the script)
- User defines just data sets to drive tests with
- Data is de-coupled from script & defined in external data source (Data pools, Excel spreadsheets etc. for data sets)
- Flow control (navigation) is normally done by the test script not via the data source

Example: data set exercises creation of new sales accounts functionality; stored in a DB table account_data

CompanyName	PrimarySalesPerson	Street	Zip	City	State
Genesis Inc.	Phil Collins	5775 Main st	30075	Atlanta	GA
RollingStones Inc.	Mick Jagger Jr.	2332 Washington st	02111	Boston	MA

Java Code Snipet of Data-Driven RFT Script

```
public void testMain(Object[] args)
{
String name;String primarySalesPerson;String address;String zip;String city;String state;
String query = "Select * from account_data";String AccessDBPath = Constants.FL_DB_PATH;

//Data abstraction class DataHandler – methods process data and return data values
DataHandler dlrAcctData = new DataHandler (AccessDBPath,query);
ArrayList listDlrAcctData = dlrAcctData.getRowsOfData();

try {
    for (int i=0;i<listDlrAcctData.size();i++){

            coName = dlrAcctData.getValueForColumnName(i,“CompanyName");
            primarySalesPerson =
               dlrAcctData.getValueForColumnName(i,"primarySalesPerson");
            Type.text(CustomerCreate.field_name(),coName );
            Click.object(CustomerCreate.image_primarySalesPersonIcon() );
            Type.text(text_employeeModalPanelCommonN(),primarySalesPerson);
            …
    }
dlrAcctData.closeDB();
}
…
```

3rd Generation Automation Framework "Keyword-Driven"

- Automation is action-centric
- Tool-agnostic (abstracted from tool if implemented correctly)
- Users de-compose test cases into granular & re-usable Action Keywords
- The idea is for non-coders to be able to author automated test cases with Action Keywords
- User defines flow control of the test via Action Keywords

Example: *Test Case "Verify Checking Account Balance"*

- *Enter* Username and Password and *Click* submit button (entire step 1 may become action *Login)*
- *Enter* "Phil Collins" as a Sales Person and *Click* Submit button
- *Verify* the Sales Person was successfully created and *Logout*

So you may want to choose the following re-usable action keywords:
Login; Enter; Click; Verify; Logout

Benefits of “Keyword-Driven” Approach

- Test Case definition is abstracted from the tool and can reside in external data sources (DB tables, Excel spreadsheets ...) or Test Case Management Tool that supports keyword based test cases (QualityManager, QualityCenter …)

- Lands itself well with Agile methodology
  - Keywords can be extended to emulate domain language
  - Customers and non-technical testers can define tests that map to business workflow more easily

Example of Keyword-driven test case:

Step	Description	Page	Action	Module	Type	Object	Expected
1	Login	Home	Login	UserLogin	N/A	. id:=LoginSubmit	.text:=Login Successful
2	Enter New Sales Person data	CreateSalesPerson	EnterText		Field	. text:=SalesPerName	.value:=Phil Collins
3	Click Submit	CreateSalesPerson	Click		Button	id:=SubmitSalesPer	url:=.*createdSalesPerStatus.html
4	Verify Sales Person Creation Successful	CreateSalesPersonStatus	VerifyExists		DIV	. id:=CreationStatus	.value:=User Created Successfully

Java Code Snipet of Keyword Library Class

```
package framework.keywords;
…
public class Verify {
…
    public static boolean exists(GuiTestObject object) {

        logInfo("Action: VerifyExists");

        try {
                object.exists();
                logInfo("Object Found");
                passed();
                result = true;
            }
        catch(NullPointerException e) {
                logError("Object Not Found");
                failed();
                }

        return result;
    }
}
```

Hybrid Test Automation Framework

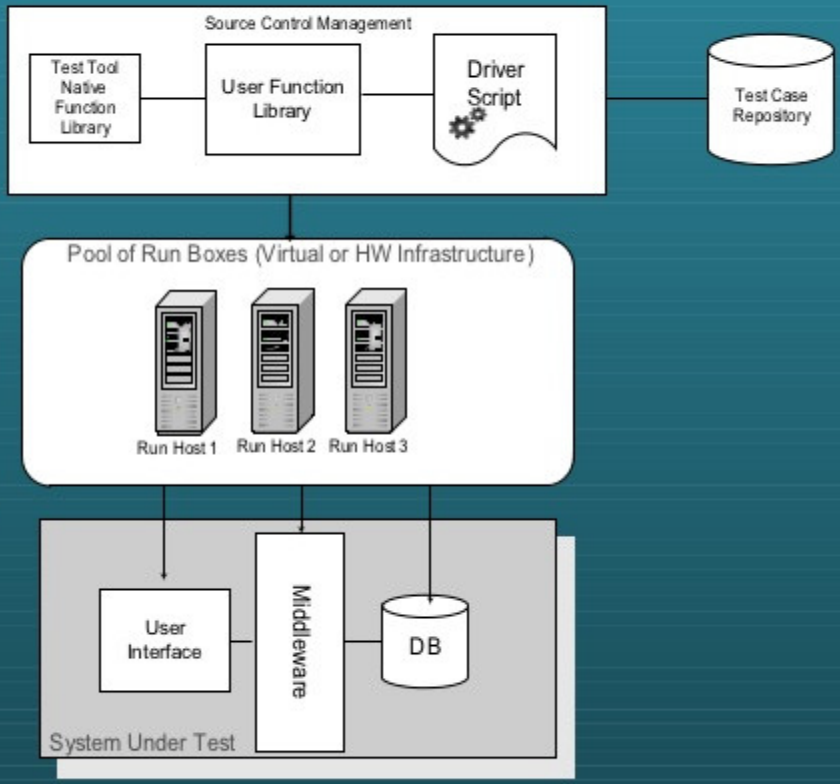
- Combines the best of both worlds (Keyword-driven + Data-driven)
  - User defines data sets to drive tests with
  - User also defines flow control of the test via action keywords
  - Data is separated from script and is stored external data source (DB tables, Excel spreadsheets, XML for data sets) with action keywords in addition to generic and test case specific data sets

Framework Type Recommendations

Framework Type	Pros	Cons
Data-driven	•Takes less time to design & implement compare to keyword or hybrid (less of initial investment)	•Flow control is not abstracted & implemented in the script Difficult to use for non-coders
Keyword-driven	•Non-coders can author automated test cases with action keywords •Makes it possible to scale Test Automation Creation & Execution •User defines flow control of the test via action keywords (Login, Click …) •Creates Domain Language •Powerful technique for Agile Testing	•Takes more effort to design & implement and may require more coding expertise •Data driven iterations may not be implemented
Hybrid	•Non-coders can author automated test cases with action keywords •User defines flow control of the test via action keywords (Login, Click …) •Users can data-drive the iterations of test execution •Data is separated from the script	Takes the most effort to evolve and thus requires the most investment with design & implementation

Test Execution Phase

- Based on personal experience 20 step UI-driven test case would take about 2 minutes to execute with 1 run box
- To scale total test execution time linearly add run boxes to the pool, e.g. with 5 run boxes of the same HW grade you can execute up to 100 steps in 2 minutes
- Now compare how long it would take a manual tester to execute 100 steps → that alone will self- justify value of test automation

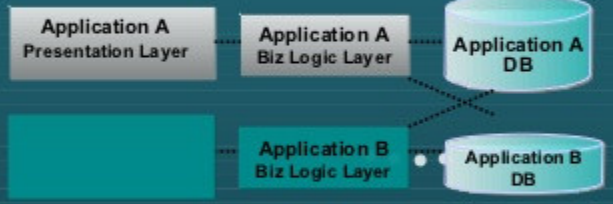


Service Oriented Architecture

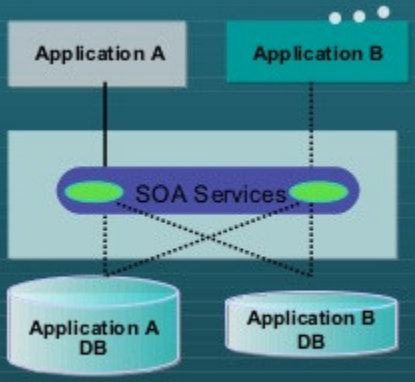
- Definition: SOA is an architectural style whose goal is to achieve loose coupling among components which leads to greater re-use of business logic

- As a result, SOA
  - **Streamlines applications**
  - **Standardizes multiple system interfaces between consumer and backend systems**
  - **Centralizes commonly used functions**
  - **Provides a communication layer between the application and backend systems**

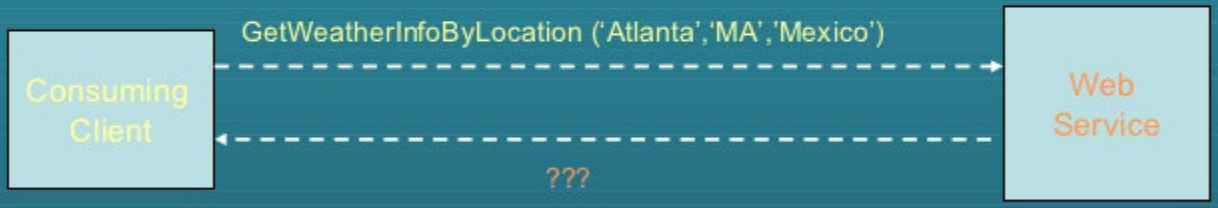
***Before** SOA (3-tier architecture)*



***After** SOA*



Web Service versus GUI-based Testing



- Web Service has no preventative input validation like the GUI
- Greater reliance on Data – data handling under positive, boundary, negative, null conditions

Impact: Single low-quality service can impact multiple consuming clients

Challenges Testing SOA

Testing Middleware/SOA is inherently more complex than testing traditional enterprise GUI-driven applications:

- Middleware is head-less (no GUI) and thus testing is similar to API
- Middleware is not built with viewer in mind hence messages require and return far more data (like XML) than human-oriented user interfaces

That means:

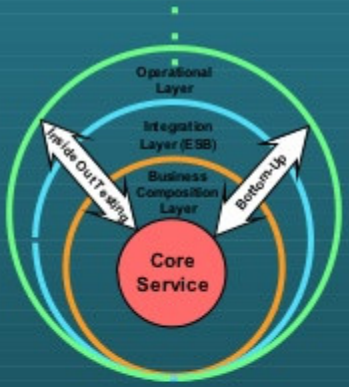
- Greater need for SOA architectural and developer skills throughout the lifecycle of testing in a SOA Tester (WS-I, WSDL, SOAP, XML, Java, .NET)
- Greater need for collaborative cross-functional test teams

The good news is:

- SOA testing will run it's own Full Life Cycle Testing independent of the application testing
- Continuous testing across the delivery lifecycle must be part of quality management strategy – automation plays greater role especially with regression and performance

Layered Testing Approach

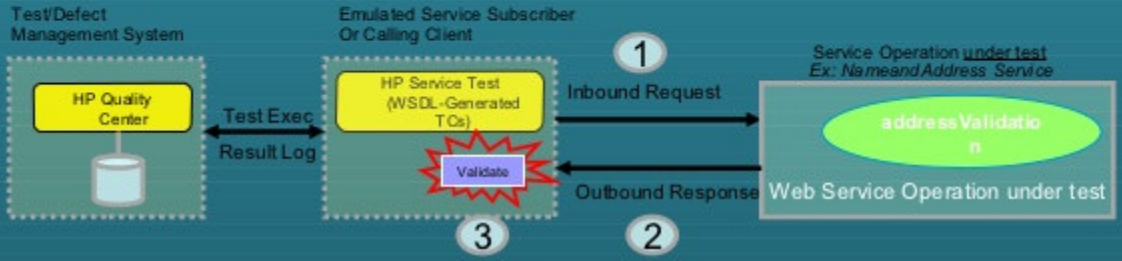
Like an onion a deployed web service has many layers



How do we approach testing complexity of web service layers ?

- A layered testing approach is also required (core component-level to integration-level to system-level testing).
- More rigorous testing with completion of proper testing at each level
- Greater reliance on automated testing

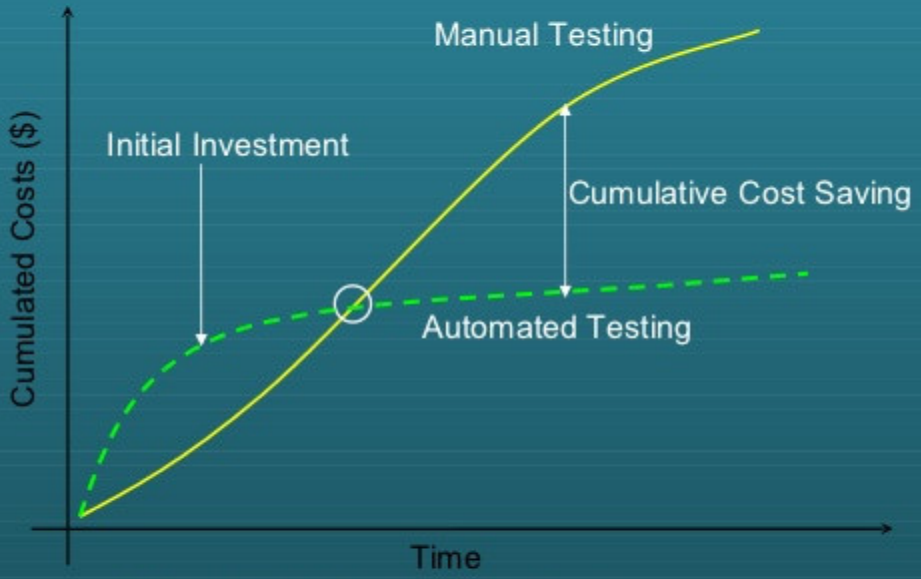
Automating Testing of Web Service Components



1. Using SOA test tool (HP Service Test) generate inbound service request based on SUT (Service Under Test) artifacts i.e. WSDL, XSD schema, design documents and familiarity of source code. Drive inbound requests parameters with negative, positive, boundary and null data values.
2. *Configure outbound response of SUT to point to calling client (HP Service test)*
3. *Response is validated via checkpoints in SOA Test Tool (HP Service Test)*

Methodology to Reduce Test LifeCycle and Accelerate Parallel Development - iTKO VSE

Calculating the ROI



ROI= Net Value / Investment Cost

* The maintenance cost of TA is usually not trivial but if you keep enhancing your TA framework, improve application testability and overall testing processes high ROI is very much achievable

Resources

- STAF – Software Test Automation Framework
  http://staf.sourceforge.net/
- Automated Testing Institute
  http://www.automatedtestinginstitute.com/home/
- LinkedIn User Groups
  - http://www.linkedin.com/groups?home=&gid=159501&
  - http://www.linkedin.com/groups?home=&gid=46748&trk=anet_ug_hm

Bibliography

- **Implementing Automated Software Testing: How to Save Time and Lower Costs** by Thom Garrett, Elfriede Dustin; **ISBN-10:** 0321580516; **ISBN-13:** 978-0321580511
- **Agile Testing: A Practical Guide for Testers and Agile Teams** by Lisa Crispin, Janet Gregory **ISBN-10:** 0321534468 ;**ISBN-13:** 978-0321534460
- http://www.stpcollaborative.com/magazine