Maximizing ROI

and

Avoiding the Pitfalls

of

Test Automation

Bill Hayduk
Founder/President
Director of Professional Services
RTTS

Maximizing ROI in Test Automation

> *the Department of Commerce's National Institute of Standards and Technology released a study late last month that puts a dollar figure on the cost of buggy software: $59.5 billion annually.*
>
> *The study … also found that, though all software errors can't be caught, an improved testing infrastructure that enables earlier and more effective identification and removal of software defects could eliminate $22.2 billion, or more than 1/3 of the money lost."*
>
> - InformationWeek

Since the implosion of the global economy, return on investment (ROI) as a measurement has again become an important means of evaluating whether a project should be undertaken. ROI, as a term representing value above expense, has become part of the business lexicon. It is incredibly useful when comparing one IT project to another or when deciding whether or not to undertake a project at all.

In this paper we will take a look at the ROI derived from automated testing, targeting both functional and scalability (load, stress, performance, volume, etc.), and try to determine guidelines for calculating the ROI.

ROI = Net Present Value of (or benefit derived from) Investment / Initial Cost

This equation may appear to be straightforward, but the difficulty lies in determining the value of intangible benefits derived from automated testing, since the effort will not directly produce revenue. We will look at ROI and test automation in broader terms, rather than in the explicit terms of the above formula and try to determine its value.

What is test automation?

Test automation is the use of test tools to robotize the exercising of business and system transactions and requirements to verify application and architecture correctness and scalability/performance.

Most automation testing tools have editors, compilers and fully functional programming languages, i.e. C, Basic, Java, or Javascript languages (see figure 1).

Warning!

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 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

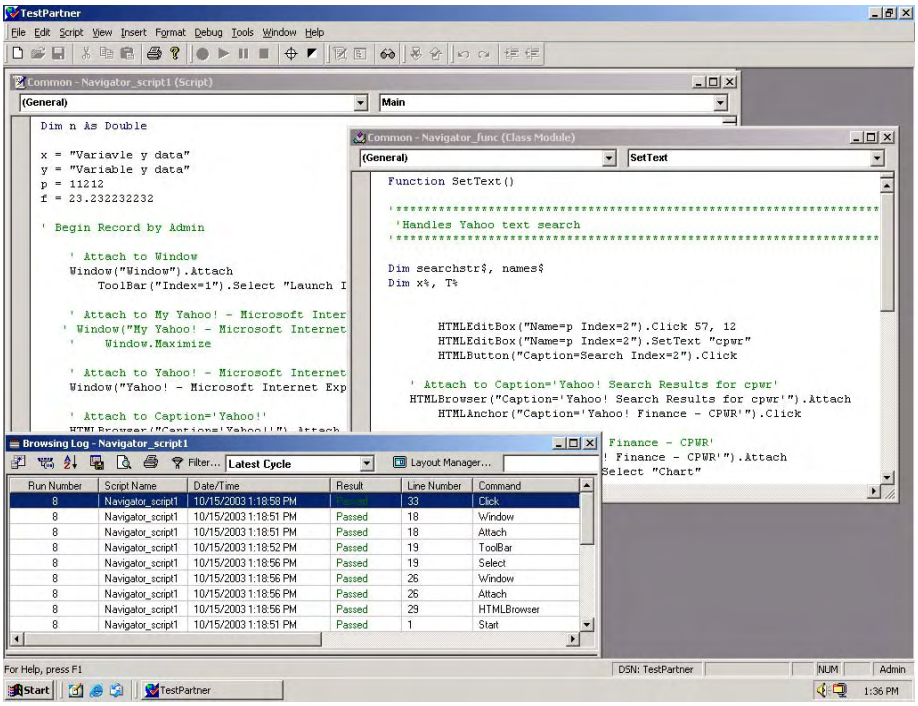

Figure 1 – the IDE, code and log of Compuware TestPartner

What are the 2 primary tools types for Testing & QA Groups?

1) Functional/Regression testing tool – This type of tool installs on a desktop and records objects and data entry and allows for playback of transactions against baseline information. It can be leveraged to provide complex testing functionality.
2) Performance/Load testing tool – This type of tool installs on a desktop and records either SQL calls, HTTP calls, socket capture (from Winsock dll) or other proprietary protocol from that desktop. Performance tools allow for playback of many concurrent users performing the same or different transactions at the same or different times from that desktop or from other servers/desktops through the use of agents.

Why run functional regression and scalability/performance tests?

Functional regression tests are run to verify that the application and architecture meets business requirements and to verify that new builds and upgrades have not impacted the functionality of the application. Scalability/performance tests verify that the application and architecture can scale to the expected user population, that the application meets service-level agreements (SLAs), and assist with fine-tuning the architecture to get it production ready.

Who are the main testing tool vendors?

The main tool vendors for performance and functional automation testing are listed below. All have their strong points and work in most mainstream environments. Each tool has target architecture (front-end development tools, protocols and databases) that their tools work best with.

Functional Regression and **Performance/Load** Tools and Vendors

These tools provide automated record and playback of functional and performance testing transactions. Both functional and performance/load tools provide full IDEs (development environments) and both can be extended using programming techniques.

Company	Web Site URL	Functional Testing Product (s)	Performance Testing Product
IBM/Rational Software	http://www-01.ibm.com/software/rational/offerings/quality/	*Functional Tester*	*Performance Tester*
HP	http://welcome.hp.com/country/us/en/prodserv/software.html	*WinRunner, QuickTest Pro*	*LoadRunner*
Compuware	www.compuware.com	*QARun, TestPartner*	*QALoad*
Microsoft	http://www.microsoft.com/visualstudio/en-us/products/teamsystem/default.mspx		*Team System: Test Edition*
Borland	www.borland.com	*SilkTest*	*SilkPerformer*
Empirix	www.empirix.com	*e-Tester*	*e-Load*
Radview	www.radview.com	*WebFT*	*WebLoad*

Code Testing Tools and Vendors

These tools are targeted at developers to evaluate code. They identify runtime or memory-access errors in Visual C/C++ , Java, VB.NET and C# code; find performance bottlenecks in app server code including Java Server Pages (JSPs) and Java Servlets; and perform code coverage, which identifies untested code quickly.

Company	Web Site URL	Product (s)
Compuware	www.compuware.com	*DevPartner*
IBM/Rational	http://www-01.ibm.com/software/rational/offerings/quality/	*Purify+*
Microsoft	http://www.microsoft.com/visualstudio/en-us/products/teamsystem/default.mspx	*Team System: Test Edition*

Security Testing Tools and Vendors

These tools provide automated Web application scanning and testing for all common Web application vulnerabilities, including WASC threat classification - such as SQL-Injection, Cross-Site Scripting, and Buffer Overflow - and intelligent fix recommendations to ease remediation.

Company	Web Site URL	Product (s)
IBM	http://www-01.ibm.com/software/rational/offerings/websecurity/webappsecurity.html	*AppScan (formerly Watchfire)*
HP	https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-201_4000_100__	*Application Security Center (formerly SPI Dynamics)*

Application Profiling Tools and Vendors

Utilizing network trace files, these tools work in a pre-deployment predictive state and in a production environment to check for network latency, chattiness of application, troubleshoot performance problems, and predict application performance over a LAN and/or WAN.

Company	Web Site URL	Product (s)
Compuware	www.compuware.com	*ApplicationVantage*
OpNET	www.opnet.com	*IT Guru*

Application Performance Monitoring Tools and Vendors

These tools monitor transaction throughput and key components of application's architecture (i.e. the memory, cpu utilization, disk i/o of client desktop, web server, app server, database server, etc.) Also, they allow for setting of thresholds and sending alerts when these thresholds are exceeded.

Company	Web Site URL	Product (s)
Compuware	*www.compuware.com*	*ClientVantage and ServerVantage*
HP	*www.hp.com*	*Business Availability Center*
IBM/Tivoli	*www.tivoli.com*	*Tivoli Monitoring*
Keynote	*www.keynote.com*	*Keynote Services*
ProactiveNet	*www.proactivenet.com*	*Proactivenet Monitor*

Application Life Cycle Quality Management Tools and Vendors

These tools manage the test effort for both manual and automated testing and provide a common area for developers, testers and business analysts to manage the lifecycle. Typically they are role-based and

Company	Web Site URL	Product (s)
IBM	*www.ibm.com*	*Rational Quality Manager*
HP	*www.hp.com*	*HP Quality Center*
TOMOS Software	*www.reachsimplicity.com*	*TOMOS Software*

What are the initial costs incurred in test automation?

There are basically four different groups of costs associated with test automation.

1. **The cost of the software**. The cost of software for functional testing costs approximately $5,000 per user. The cost of a performance testing tool can run from $35,000 for 500 virtual testers and only the http protocol (web) up to $300,000, depending upon features, number of virtual testers and architecture (and protocol) to be tested.

2. **The cost of the hardware.** The cost of hardware is negligible for functional testing. A high-end workstation can be purchased for under $2,000. By high-end, we mean a 3-ghz processor, 2+ gigabyte of RAM and an Ethernet 1 GB port. To implement functional automation, a workstation will be needed for each engineer.

   The cost of hardware for performance/load/stress testing is significantly higher. Typically, when speaking about any type of multi-user test, tool vendors speak in terms of virtual users and virtual testers. A virtual user is a process or a thread that can emulate a transaction whereas the middle components and the backend database cannot tell the difference between the virtual user and an actual user.

   Multi-user testing (commonly referred to as 'performance testing' by the tool vendors) requires a master machine to act as the scheduler and coordinate the tests and agent machines to drive the virtual user scripts. Virtual users consume computer resources. A virtual user can consume between 300k to 6 megabytes of RAM. Also, CPU saturation typically comes at approximately 250 virtual users. Therefore, 1 high-end PC with 3-ghz processor and 2 gigabyte of RAM may be able to push 250 virtual users. The cost to purchase this hardware would be less than $2,000 to run a 100-user test (source: Dell) and $337,000 for a 5,000-user test (source: Sun). Some software vendors and professional services firms provide performance testing software and the hardware to run the virtual users on a rental basis, greatly reducing the cost per project (as does RTTS).

3. **The cost of trained personnel.** Either training and mentoring of an internal resource ($75,000 for salary, benefits, plus $25,000 for training, mentoring and experience over the course of 2,000 hours to become proficient)

or contracting a skilled resource from a professional services firm (from $500 to $2,500 per day, depending upon which firm is contracted and the skill set and experience of the resource).

4. **The cost of scripting (or coding) the test cases.** On the functional side the cost is the up-front time for setup and the fact that scripting test cases takes five times longer than manual testing in the initial startup period. (Hence the break-even point on functional test automation is said to be at least 5 anticipated builds).

What are the tangible benefits of automated testing?

- **Speed and Accuracy** – It's faster and more accurate than manual testing. It can be as much as 50 times faster, depending upon the speed of the driver machine and the speed of the application to process information (inserts, updates, deletes and views). Test tools also are much more accurate than manual test input. The average typist makes 3 mistakes for every 1,000 keystrokes. Also, automation tools never tire, get bored, take shortcuts or make assumptions of what works.
- **Accessibility** – Automation tools allow access to objects, data, communication protocols, and operating systems that manual testers cannot access. This allows for a test suite with much greater depth and breadth.
- **Accumulation** – Once tests are developed, long-term benefits are derived through reuse. Applications change and gain complexity over time. The number of tests is always increasing as the application/architecture matures. Engineers can constantly add onto the test suite and not have to test the same functionality over and over again.
- **Manageability** – Ability to manage artifacts through automation tools.
- **Discovery of issues** – Automated testing assists with the discovery of issues early in the development process, reducing costs (see figure 5 below).
- **Repeatability** – An automation suite provides a repeatable process for verifying functionality on the functional side and scalability on the performance side.
- **Availability** – Scripts can run any time during the day or night unattended.

What are the intangible benefits of test automation?

- **Formal process** – Automation forces a more formal process on test teams, due to the nature of the explicitness of the artifacts and the flow of information that is needed.
- **Retention of customers** – When sites do not function correctly or perform poorly, customers may leave and never come back. What is the cost to your business of that scenario? Performing correct and systematic automated testing helps assure a quality experience for the customer – both internal and external.
- **Greater job satisfaction for Testers** – The Test Engineers no longer manually execute the same test cases over and over. They would utilize a programming-like IDE and language that is more challenging, rewarding and portable to other positions (ie development).

What is a rule of thumb for determining whether there is sufficient ROI to undertake functional test automation?

When testing the functionality of an application, the heuristic (or "rule of thumb") is whether there will be at least five builds. For scalability/performance testing, any application with more than a "handful" of concurrent users on a site that is critical to either internal (employees) or external customers.

Below are the primary reasons for failure in automation. We have listed them below and provided RTTS' solutions for overcoming them.

1. Lack of structured automation methodology.
2. Test automation is not treated as a project with proper project planning (i.e. scope, resources, time-to-market).
3. Testing is performed at the end of the development cycle (the waterfall method).
4. Approaching all architecture from the user interface (black box testing).
5. No modularization (use of functions) in automation scripts.
6. Test engineers are untrained in tool interface and programming techniques.
7. After initially creating automation suite, customer does not maintain the suite for future builds.

Below are listed the solutions to the reasons for failure.

1. *Implement a structured automation methodology.*

Implement a pragmatic approach to testing that is *manageable, repeatable, measurable, improvable, automated* and *risk-based* (see figure 2).

Manageable, such that the project can be decomposed into modular, defined tasks with assigned resources and timelines.

Repeatable, such that others can easily carry forward the process that has been defined.

Measurable, such that the effort is quantifiable - how many defects found in each stage, what is trend of different severities of defects, how close is the testing cycle to completion, how long does a transaction take to complete?

Improvable, such that each build becomes more efficient in producing defects. The goal of this measurable and improvable process is to produce more defects in the testing life cycle so that less are found in production.

Automated, to build a data-driven regression and scalability/performance suite that takes advantage of the best-of-breed testing software.

And *Risk-based*, by targeting test types and application functionality that is the most crucial to the usage of the application(s). RTTS implements *TAP*, it's Test Automation Process, defines tasks, heuristics, project tracking, reports, metrics and "what-if" scenarios over a distributed test model (see figure 3).

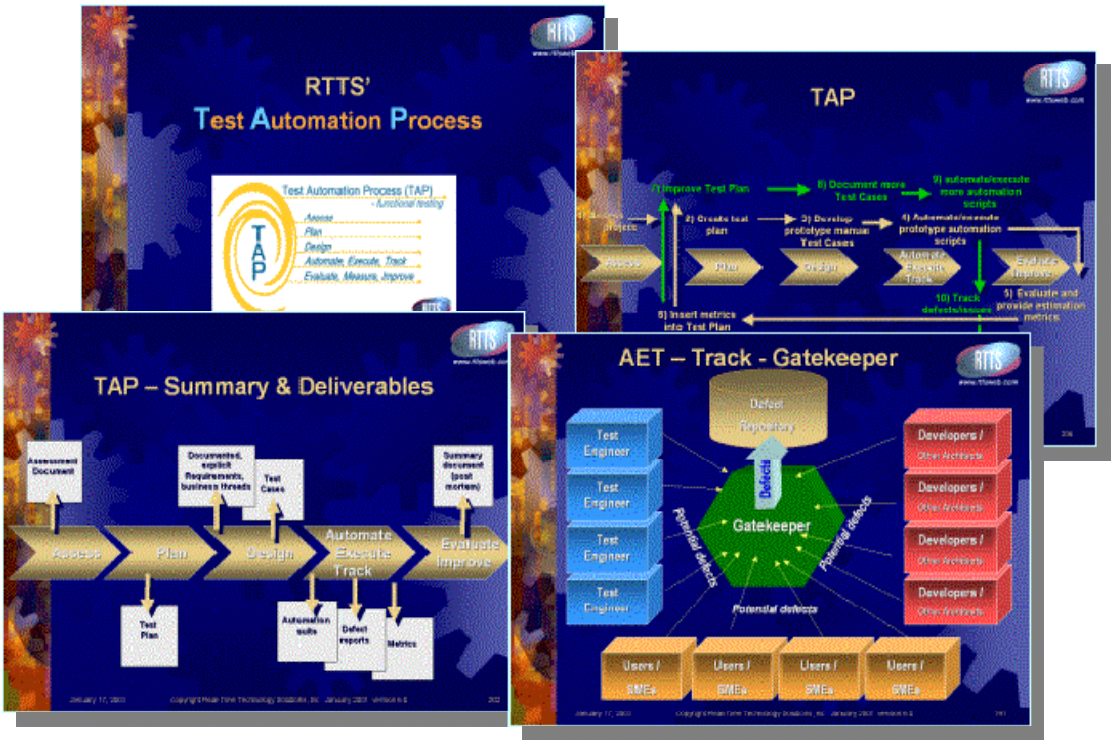

Figure 2 – RTTS' Test Automation Process (TAP)

2. ***Treat testing as a project.*** Effectively treat test automation as you would a development project and manage the scope, resources and time-to-market adequately. The three variables have interdependencies (see figure 3). Since both resources ("I only have budget for x testers.") and time-to-market ("If we don't deliver this software by x date, we're all out of jobs.") are typically not variables, but constants, the only component that is truly a variable is scope. Only "z" scope can be tackled by "y" resources in "x" time. If "x" and "y" are fixed and the scope is greater than what y resources can perform in x time, then not all of the functionality can be tested. And if the scope of the effort needs to be increased, there are 2 choices: increase the resources or extend the time frame. The same holds true if the scope of work required exceeds the amount of work that can be performed by the resources in the time frame available – the scope must be decreased.

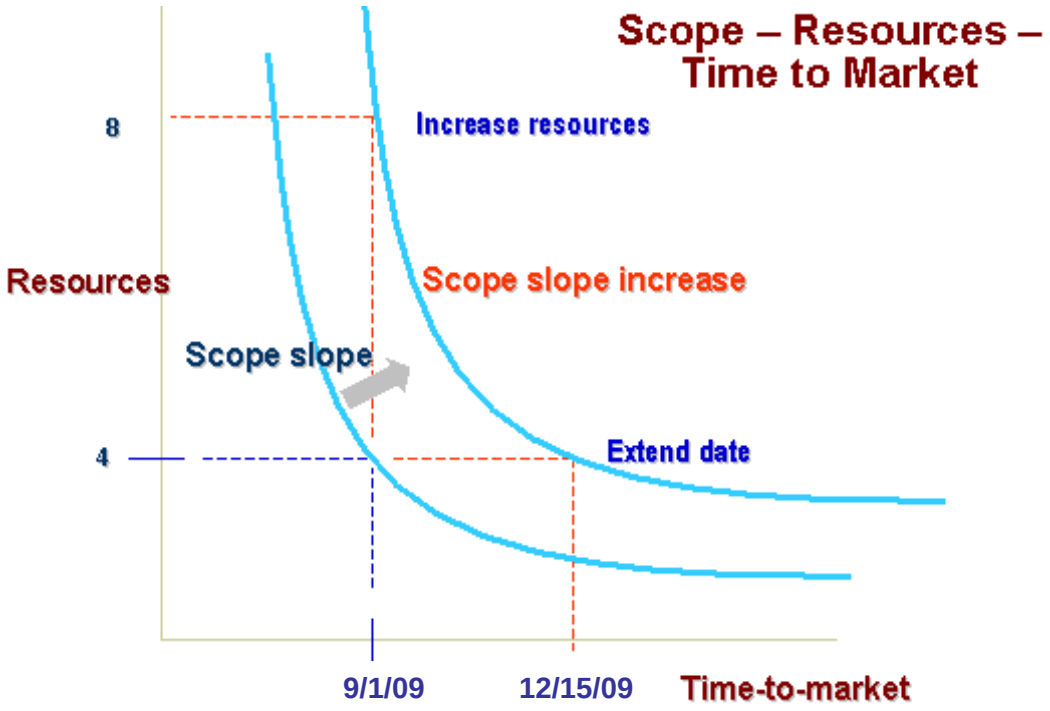

Figure 3 – management of the 3 project variables

3. ***Move testing up in the software development lifecycle.*** The test process should begin where the development process does, at the beginning. Some development teams still follow the Waterfall development process (see figure 4), which dictated that testing was done in stage 5 and was 10% of the entire development effort (Gartner suggests 30 –40%). This process was well-suited to the stability of mainframes, mainframes, but is ill-suited to complex, multi-tiered iterative system development. Defect detection proves much too costly (see figure 5). Moving the test process up in the software engineering cycle minimizes the cost of defects and provides more time for effective test planning, design, execution and tracking.

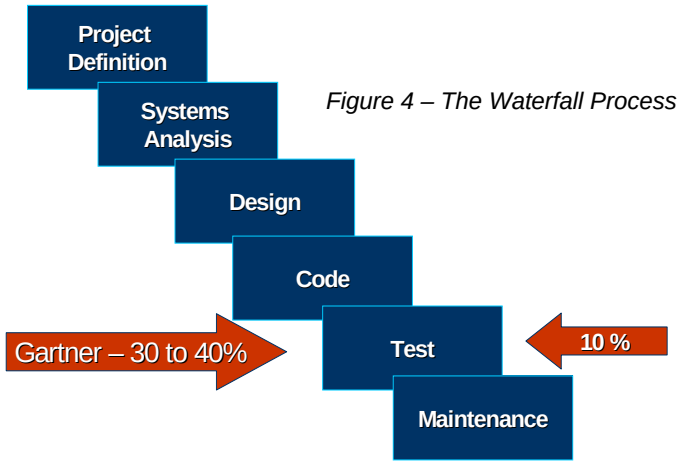

Figure 4 – The Waterfall Process

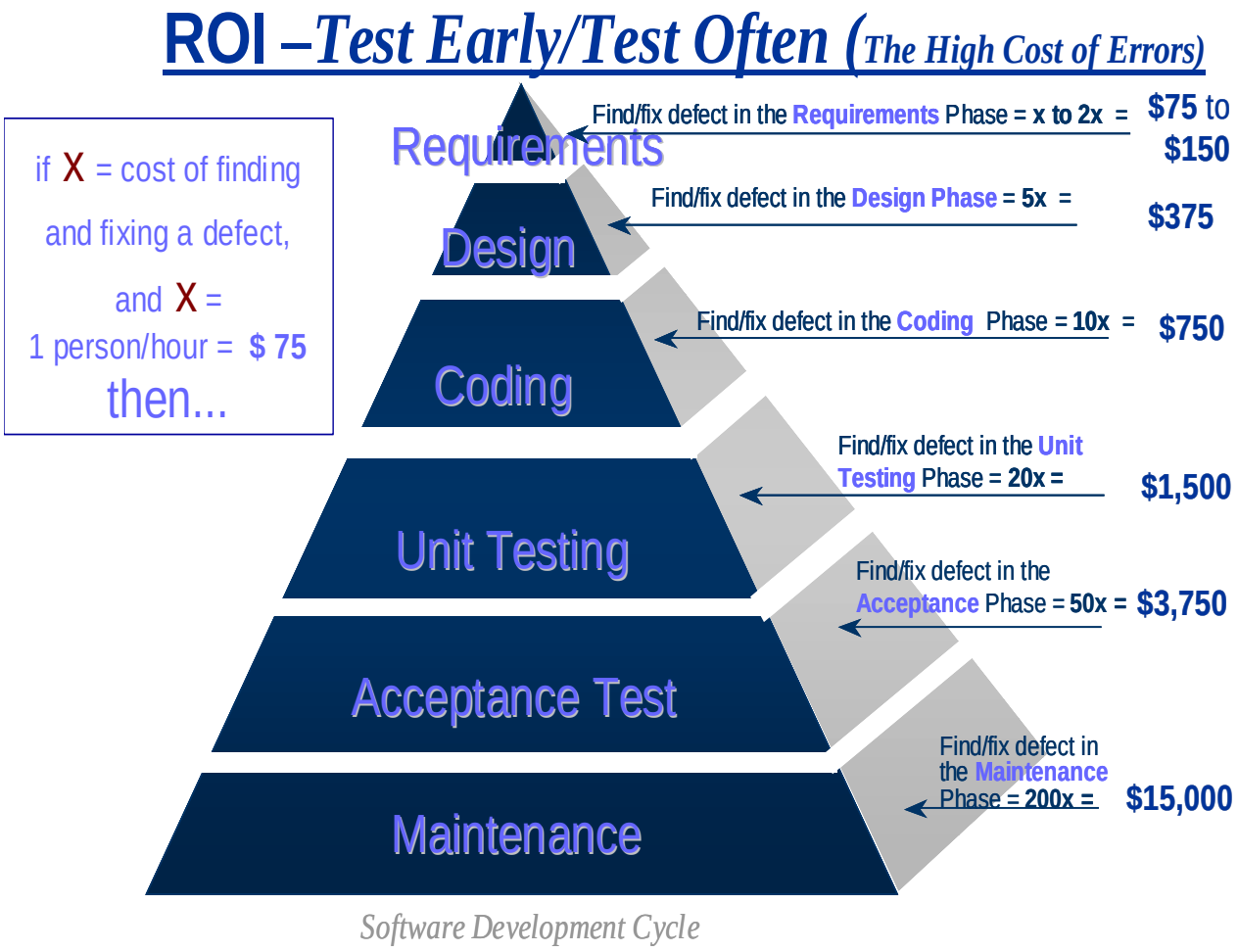

Figure 5 – The high cost of defects

4. ***Provide a component-based testing model.*** During the early to mid 1990's, when 2-tier client/server architecture became prevalent, automation tools for the distributed environment burst onto the scene. These tools provided a great way for functional regression and scalability/performance testing through the user interface, or from a black box perspective. If the issue found was not on the client side, then it was on the server. With today's complexity of heterogeneous architecture (see figure 6), it is no longer satisfactory to say that the application does not function correctly or cannot scale. Automation engineers now need to develop a strategy that helps determine where the issue lies. This requires a component-based approach where the application architecture is decomposed into smaller components. Data is traced and verified from tier-to-tier and servers are monitored while load is ramped up. This strategy will answer not only whether (1) the application functions correctly and (2) can scale but also *where the incorrectness in functionality resides* and *where the bottlenecks are*.

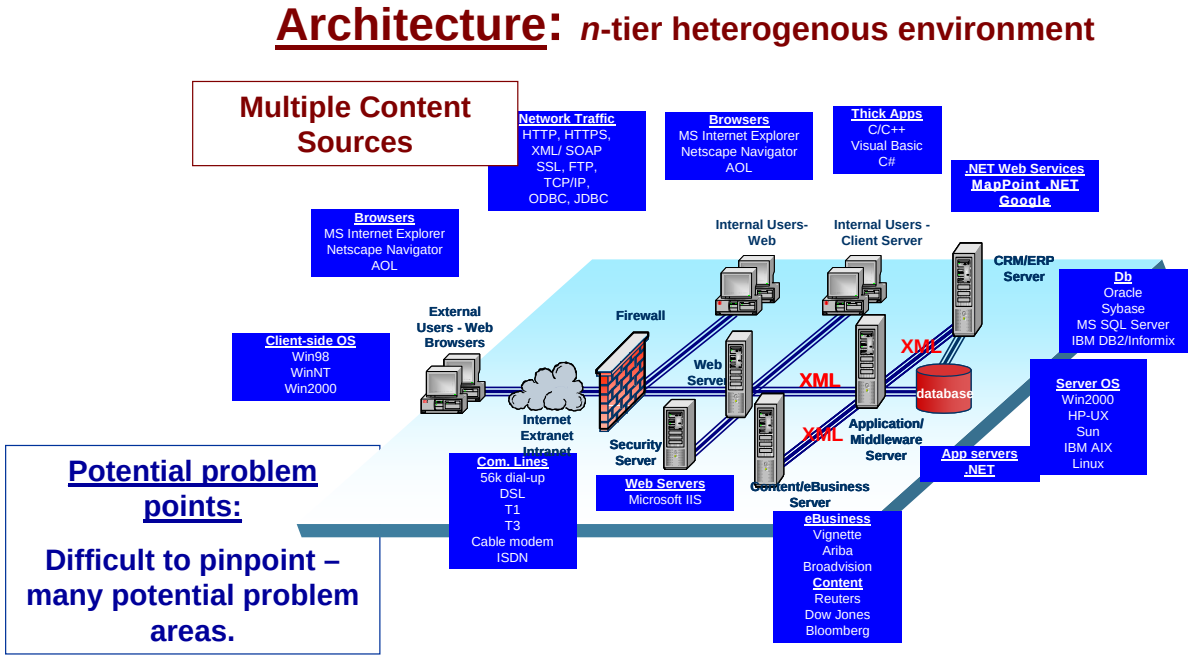

Figure 6 – Complex, heterogeneous architecture

5. ***Modularize the automation scripts.*** Utilize an effective test automation coding strategy (with automation tools) of wrapping redundant navigation, data input and baseline verifications into function libraries, which modularizes scripts, prepares the scripts for changes to the application and minimizes the maintenance of scripts of future builds (see figure 7).

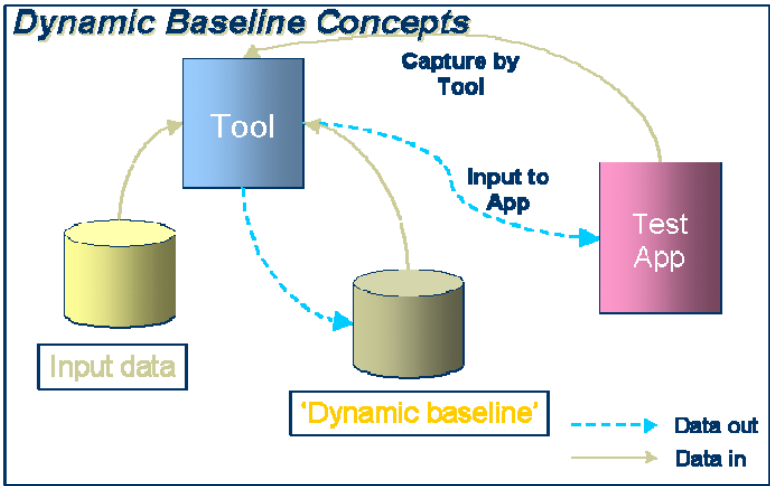

Figure 7 – Dynamic baseline scripting

6. *Train engineers in testing and test automation discipline.* Develop a hiring, training, and support model that includes engaging the appropriate resources, developing a rigorous training program and a building a successful mentoring/support model (see figure 8). It typically requires 6 – 9 months of targeted lessons, workshops, labs and assignments to fully train a functional test engineer and 9 – 12 months to fully train a performance test engineer.

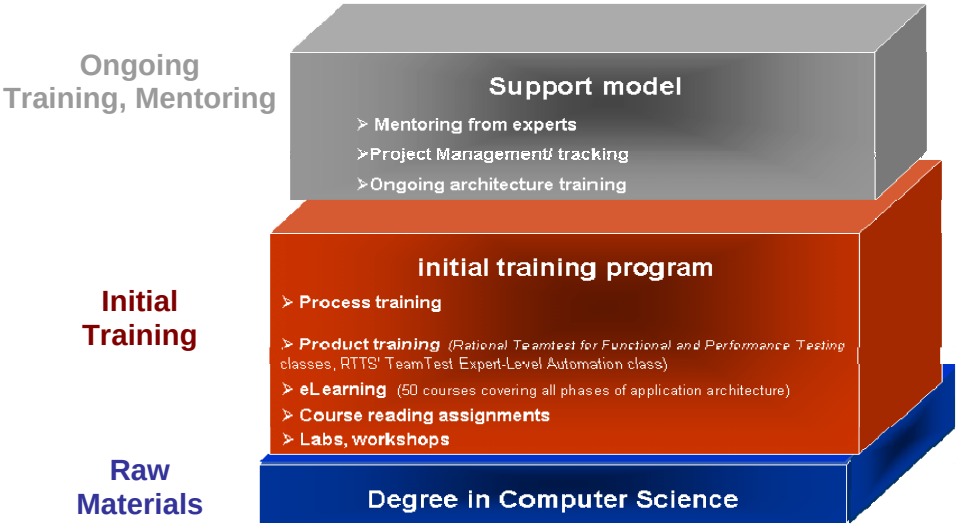

Figure 8 – RTTS' training and support model

7. *Maintain your test suite.* Why spend all that time (and money) developing a robust suite of automation tests and then not support and maintain it? Test suites need to be maintained with each new build and release of an application. Plan for a script maintenance program by either creating a centralized test team that performs the work (see figure 9) or contracting out a firm who provides this service. Maintenance of robust scripts typically requires 20% of the time of originally creating the automation scripts (assuming that major overhauls are not being done to the system-under-test). By utilizing a centralized team or utilizing an outsourced service for maintenance, resources can be allocated more efficiently and your investment can be protected.

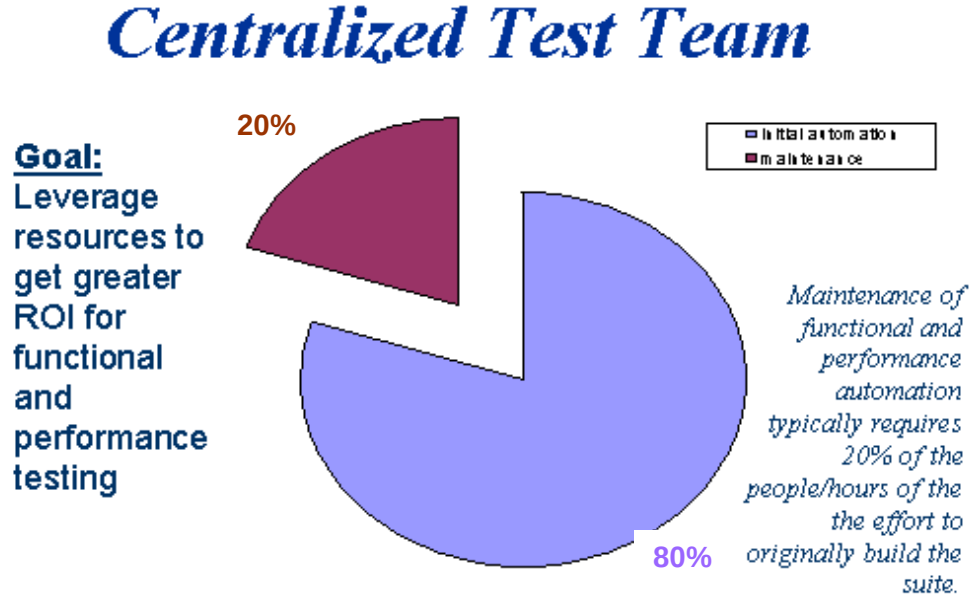

Figure 9 – Script Maintenance as a percentage of overall scripting

Case Studies

ROI - Case Study – functional testing

Industry: Insurance

Business Purpose: Verify the functionality and validation of the base system for entering life insurance policies.

Number of Automation Scripts: 3,900

Time to run automation scripts: 12 business days

Initial cost: 557 people /days (2 engineers working concurrently built this suite)

Time to manually execute: 139 people/days (28 per day)

# of runs to break even: 4.01 runs (to date there have been > 15 runs)

ROI: Savings of time (127 business days on each run), improved customer satisfaction (126 critical defects found and addressed).

Details: This Fortune 500 client found automation so beneficial, they created a centralized testing group to leverage automation throughout their organization. They implemented load, volume, stress and performance testing, along with rigorous functionality testing. Testing has become an integral component to delivering to their internal customers high-quality products.

ROI - Case Study – scalability/performance

Industry: Brokerage

Business Purpose: Scale on-line brokerage application to 20,000 concurrent users

Number of Automation Scripts: 22

Time to perform 1 set of tests: 1 hour per run (1-user, 100 users, 500 users, 1,000 users and then incrementing by 1,000 until 15,000 users)

ROI: Found critical system errors in architecture components, database issues. Multiple iterations allowed for a much more scalable architecture and application, enhancing the user experience.

Details: This customer's success also led them to form a centralized function for load and performance testing. Due to the large volume of transactions that they rolled out to their very large client base, their test team was responsible for pinpointing bottlenecks and problematic areas of scalability before these applications went into production. These tests were a pre-emptive solution for providing quality software that scales to meet the performance requirements of a customer base that has many available competitive choices, should the online systems not meet the customers' needs.

Example of Calculation

Let's assume you are designing software to be run on your clients' desktop. The architecture is web on the client side performing complex transactions. To appeal to the greatest market, you have decided to support Internet Explorer 7.x and Firefox 3.x, Windows 2000, XP and Vista. Let's assume you have identified 500 scenarios (transactions) that test functionality and verify data throughout the system. Your choice is manual testing versus automation. (We will assume that the choice 'not to formally test' is illogical because it would only push large quantities of defects to production- i.e. your clients, and eventually force them to go to another vendor, effectively putting you out of business.)

Item	Manual Tester	Manual Cost	Automated Tester	Automated Cost
Cost	$75 per hour		$75 per hour	
Time to author 1 test case	1 hour per or 7 per day		1 hour per or 7 per day	
Time to author 500 test cases	71.4 days	$42,840	71.4 days	$42,840
Time to automate	0 days	0	100 days	$60,000
Time to execute and analyze EACH test case and results	3 per hour or 24 per day		20 per hour or 240 per day (depending upon the speed of the processor and the speed of the application)	
Time to execute and analyze ALL test cases and results	20.8 days	$12,480	2.1 days	$1,260
Total run through the code for 1 release		**$55,320**		**$104,100**
Time to test for each subsequent release	20.8 days[1]	$12,480	Rework on scripts at 5% or 5 days[1]	$3,000
Assume new code releases 1x per month or 12 per year	249.6 days	$149,760	Rework on scripts at 5% or 60 days	$36,000
Total for 1 year		$205,080		$140,100
Cost of testing on all pertinent combinations (IE & Firefox on Win 2000, XP and Vista)	All tests run on 6[1] configurations for each code change.	**$953,880**		**$320,100**

[1] Assumes no additional test cases are authored or automated.

The savings in this example would be a tremendous time savings, which would equate to $633,700 .

Evaluate your application as a candidate for automation….

IF you answer **YES** to any of these questions, your application is a good candidate for automation based upon a return on investment.

- Does your application need to be tested on multiple hardware or software configurations?
- Will you have more than 5 new builds/patches/fixes of the application?
- Is a vendor developing the application for you and needs to meet Service Level Agreements?
- Do you have more than 5 concurrent users on your application?
- Do you have repetitive tasks that are performed to maintain an application (i.e. data loading, configuration, etc.)?

Conclusion:

The return on investment for test automation is quite obvious. If performed correctly, the automation suite will prove to be much more efficient than manual testing in finding defects on the functional side and the only way of finding scalability/load/performance issues on the multi-user side. It can be run at night, on weekends and holidays and can be left unattended. The tools never get bored or tired and never assume the application/architecture works. And it can emulate as many users as can be anticipated accessing the application, performing any mix of transactions needed.

Therefore, the ROI, (tangible + intangible benefits/initial cost) of automated testing provides a tremendous return, as long as the 7 reasons for failure are overcome.

About the author

Bill Hayduk, founder, president and director of professional services, has an excellent reputation in the technology field and is particularly noted for his test methodology and automation expertise. For the past 25 years, Bill has successfully implemented large-scale automation projects at many Fortune 500 firms. He has managed projects in most verticals, including banking, brokerage, multimedia, ISVs, government, telco, healthcare, education, pharmaceutical and insurance.

Prior to working in test automation, Bill spent 3 years as a management consultant. Bill advised on a global bank's business process reengineering effort of their global trade unit and was also a key member of a special project core group working on one of the world's largest insurance firm's conversion from a mutual insurance company to a publicly traded firm.

Previously, Bill spent 9 years in the foreign exchange industry in various senior positions, culminating in Vice President/Chief Operating Officer of a foreign exchange trading firm where he managed the firm's trading, sales, marketing, technology, operations and finance.

Bill holds a Master of Science degree in computer information systems from the Zicklin School of Business (Baruch College) and a Bachelor of Arts in Economics from Villanova University. He has been a selected speaker at industry-specific trade conferences, as well as a source of information for corporations and has been referenced in many industry trade publications.

About RTTS

RTTS is a professional services organization that specializes in the testing of IT applications and architecture. With offices in New York, Philadelphia, Atlanta and Phoenix, RTTS has been serving Fortune 500 and mid-sized companies throughout North America. Drawing on its expertise utilizing best-of breed products, expert test engineers and proven methodology to provide the foremost end-to-end solution, RTTS ensures application functionality, reliability, scalability and network performance.

What differentiates RTTS from other professional services firms?

- *Total end-to-end testing of your application and architecture*
  Not content to "black box" test applications from the user interface, RTTS tests from the front-end web or fat client through the network, from the firewall to the security server through the web server, application server, eContent/eBusiness and ERP/CRM servers all the way to back-end database, testing over SQL, XML, SOAP, HTTP/HTTPS and other protocols.

- *Strategic Partnerships with best-of-breed vendors*
  RTTS is a consulting and training partner of *IBM, HP, Microsoft and Compuware* and supports tools from about 95% of the test tool marketplace, including both commercial vendors and open source.

- *Successful Client Engagements*
  RTTS has successfully completed 350 testing projects at 150 client sites around North America. Approximately 50% of our business is repeat customers, whose expectations were exceeded in previous engagements and came back to us for additional services. Our client list reads like a "who's who" of the business world and our success rate is unparalleled.

- *A proven, traceable test automation process*
  RTTS has developed *TAP*, RTTS' best practices framework for implementing and tracking large-scale testing of applications and architecture and integrates with traditional methodologies as well as RUP and Agile, lightweight processes.

- *Expert Testers*
  RTTS engineers possess expertise in complex web, client/server, and heterogeneous architectures, all have many years of experience implementing functional, performance, load, stress, security, volume, interoperability, component-based testing and other test types and all RTTS engineers are employees of the firm (no contractors).

To learn more about RTTS, visit www.rtts.com