METHODS & TOOLS

Global knowledge source for software development professionals **ISSN 1023-4918**

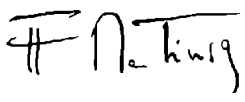
Winter 2000 (Volume 8 - number 4)

Internet Times... They Are A-Changin'

Internet time is a recent concept of our world, similar in its meaning to the theory that one year for a dog is the equivalent of seven human years. While it could also be true that life in an Internet's company is sometime very close to a dog's life, you have to admit that this concept is true. It is validated by the speed at which Internet related companies are dying! The desolation is visible in the dotcom companies area, but also in the e-consulting firms like MarchFirst, Scient, Razorfish or iXL. The wide impact of this decline indicates that the Internet enthusiasm is becoming mild, also in "old economy" organisations.

It is clear that the Web has bought new advantages to the business. It is easier to have a broader market coverage, the interactions and the feedback with the customer can be improved and you can react quicker to changing market conditions. There are some difficulties of the "mortar" world that have not disappeared with the Internet. For instance, you still have to manage logistics to deliver durable goods to the customers and you have to make yourself known by the customer. This is why one of the biggest new source of advertisement for traditional media has been the dotcom world.

These remaining difficulties explain why the bigger "successes" have been achieved by business or people that already mastered part of the issues related to e-business and that have transferred to the Net activities previously supported by other technologies (paper, phone, automated teller, country-specific networks like France's Minitel). Media, catalogue sales, customer service are activities that have found their place on the Net... with the additional games and sex sites. Amazon's books business is making profit, not its division selling domestic appliance over the Net! At the company level, the main usage of the Internet is also the evolution of "old" concepts. The current hype about B2B, XML, vertical portals, marketplaces, etc. is the evolution of the Electronic Data Interchange (EDI) trend. Intranet is only replacing something like Lotus Notes with Web sites. Now that the fad is over, Internet has to be seen for what it really is: a set of evolved technologies that can ease interactions inside and outside the organisation with its customers and suppliers. Let's put them to work!



Inside

Risk-Based E-Business Testing - Part 2 Test Techniques and Tools

Paul Gerrard, paulg@evolutif.co.uk Systeme Evolutif Ltd. www.evolutif.co.uk
9 Cavendish PlaceLondon W1M 9DL, UK
Tel: +44 (0)20 7636 6060, Fax: +44 (0)20 7636 6072

This is the second part of the second part (recursive editing?) of the article of Paul Gerrard about e-business testing.

5. Non-Functional Testing

5.1 Configuration Testing

Configuration testing aims to demonstrate that your Web application will operate correctly on your nominated range of client hardware, operating system and browsers software combinations. On an Internet, you have no control over the end users' platform, so to avoid problems later, it is best to test a range of configurations to ensure that you can support at least the most common combinations.

The problem of course, is that there are virtually an unlimited numbers of configurations available:

O/S Platforms	DOS, Windows 3.1, 95, 98, NT 3.51, NT 4, Macintosh, Unix.
Network connections	Dial-up modems, direct Internet lines, ADSL, Wireless.
Commercial services	MSN, AOL, Freeserve and many others.
Browsers	Internet Explorer, Netscape, and many others, too many to mention…

The problem of test scope

- Which configurations will you support? (Not all will be worth considering).
- Which can you test economically?
- Which can you test automatically?

If you intend to perform configuration testing on more than a handful of platforms, the first task is to analyse your market. What kind of hardware and software are your users most likely to have? One might reasonably expect your analysis to result in a small subset of what is available: MSIE and Netscape on Windows 98 and Windows 2000 might cover the majority of business-to-business clients.

But… what about Unix and Macintosh users? Microsoft NT and Windows 95 are likely to exist for some years yet, aren't' they?

If you are to look seriously at configuration testing then you must also look at component commonality. Is there any reason to doubt that one browser version will work differently on two operating system versions? If there is, what is the potential impact? Are you using any of the O/S specific features?

There are two discussions to have:

With the marketers: what are the most likely platform combinations to be used?

With the developers: are there any platform specific features being used that make configuration testing essential?

In our experience, only the largest, most mission critical projects take configuration testing so serious that they test more than a handful of configurations.

We have not the time.
Is there an Alternative?

We recommend that if you are concerned with only the two leading browsers, IE and

Netscape then you consider the following alternative:

- Identify browser compatibility problems using the static analysis tools that are now widely available. To the degree that they identify HTML features that are non-standard and not compatible to one or more browser versions they are probably more effective than dynamic testing.

- Encourage your developers and testers to use a variety of browsers and versions. In this way, one would expect that in the day-to-day operation of the Web application, compatibility problems would be detected during development or other testing.

If you are only concerned about the two leading browsers, anything that gets past these two hurdles it likely to be so obscure, that few users would every be affected and there's no guarantee that you would have found the problem anyway.

5.2 Usability Testing

The importance of usability (and usability testing) has never been as important as it is now. Typical Web site users have low boredom, frustration, inconvenience and insecurity thresholds. Home-based users may never have been trained in the use of a computer, let alone browsers and your Web applications. Regardless of sophistication, if your Web application doesn't allow the user to make enquiries and place orders easily, quickly and reliably, the site will fail. If the user cannot easily understand from your Web site how to proceed, the user will leave your site and go elsewhere.

Think about this. In terms of difficulties for users, usability is probably the first problem that they experience, usability is probably a more important requirement than your site's functionality.

So the pressure to make the user experience as easy and fast as possible is considerable.

The problem with usability is the perceived difficulty in expressing requirements. However objective requirements can be defined:

- Messages to users will be in plain English.
- Commands, prompts, messages will have consistent meanings.
- Explanations are meaningful and relevant.
- The user must always know what state the system is in.
- Noticeable and informative feedback.
- The system will help (not hinder) the user:
  - don't need to confirm limited choice entries
  - system must provide defaults when applicable
  - must not prompt for data it does not need
  - must only display informational messages as requested by the user.

These requirements can be positively identified and measured.

Web conventions matter

Web page authoring guidelines provide checklists for Web conventions. These guidelines are generally accepted good practices. In principle, all Web sites should consider committing to following them to ensure their Web sites are accessible to all users. The best thing about such guidelines is that they can be used as objective requirements. When you test, failure to adhere to these guidelines can be raised as incidents.

Jakob Nielsen's book (reference 12) is a thorough treatment of good and bad practices in Web design with lots of examples of both. (As a matter of interest, Vincent Flanders' www.websitesthatsuck.com page is both

entertaining and educational.)

Heuristic evaluation

Heuristic evaluation is a sophisticated sounding name for a very straightforward technique. Reference 18 gives a very clear description of this popular approach. The process is very similar to a peer review:

- Select your evaluators to conduct private assessment.
- Compare the user interface to good practices (the heuristics).
- Raise observations of poor usability, frustrations etc.
- Evaluators may be asked to follow scripted scenarios e.g. make an enquiry, place an order etc.
- Convene a meeting to review the issues raised and discuss possible improvements. These meeting follow a 'focus group' approach. They are semi-structured discussions that are documented.

A thorough treatment of usability design and assessment approaches can be found in reference 20.

5.3 Performance Testing

Performance and associated issues such are resilience and reliability appear to dominate many people's thinking when it comes to non-functional testing. Certainly, everyone has used Web sites that were slow to respond, and there have been many reports of sites that failed because large numbers of people visited sites simultaneously. In these cases, failures occur because applications are undersized, badly designed, unoptimised and inadequately tested.

In some respects, performance testing is very easy. All that is required is to simulate a number of users doing typical transactions and there are many tools available to do this. Simultaneously, some test transactions are used to measure response times, as a user would experience them. There are a large number of tools available to execute test transactions and take measurements. In practice, with a slow site, you can do this task manually, using with a stopwatch to take response times.

What's all the fuss about – it sounds simple. In this section, I'm not going to provide a detailed description of how performance tests are planned, constructed, executed and analysed. I refer you to reference 7, which provides a broad description of client/server performance testing. In this section, I'll discuss some key points and some issues that distinguish Web performance testing from 'traditional' client/server.

Scope and Objectives

In principle, performance testing aims to demonstrate that:

- The system functions to specification with
- acceptable response times while
- processing the required transaction volumes on
- a production sized database.

Performance testing can be very expensive so we normally advocate setting other objectives to be achieved using an automated test framework and we normally include the following aspects, in defining test objectives:

- To assess system's capacity for growth – to determine by what margin the performance of a system meets requirements and speculate by how much loads can increase before response times become unacceptable.
- Stress tests to identify weak points – to subject the system to excessive or extreme loads to find the weakest components, so they can be tuned, upgraded and the system made more resilient.
- Soak, concurrency tests over extended periods to find obscure bugs – to exercise

the system over 24 or 48 hour period to ensure that the system can be operated successfully for an extended period.

- Test bed to tune architectural components – we can reuse tests to progressively measure the effects of tuning or upgrade activities on system performance to ensure we have a fully optimised system.

Pre-requisites for performance testing

We normally specify the following pre-requisites before performance tests can be executed and the results produced deemed to be reliable. These are all explained in the paper referenced above, but there are particular difficulties in E-Business environments.

- Quantitative, relevant, measurable, realistic, achievable requirements.
- Stable software system.
- Actual or comparable production hardware.
- Controlled test environment.
- Tools (test data, test running, monitoring, analysis and reporting).
- Process.

Load requirements may be suspect

The first problem is that of workload requirements. With an internal system, there is usually at least, an upper limit on the size of load that the system under test could be subjected to. With Intranets, it may still be possible to settle on a predicted workload to subject the system to that has a rational basis. With Internets, however, there is no reasonable limit to how many users *could* browse and load your website. The calculations are primarily based on the success of marketing campaigns, word of mouth recommendations and in many cases, luck.

In these circumstances, it is perhaps better to think of performance testing, less as a test with a defined target load, but as a measurement exercise to see how far the system can be loaded before selected response times become unacceptable.

In our experience of dealing with both established firms and dotcoms, the predicted growth of business processed on their Web sites is grossly overestimated. For the purpose of acquiring venture capital and attention in the market, this is the game that is played. For performance testing, it can dramatically increase the cost of testing. Ambitious targets for on-line business volumes require expensive test software licenses, more test hardware, more network infrastructure and more time to plan, prepare and execute.

Decreased timescales may be a problem

The second obvious problem with E-Business performance testing is the time allowed to plan, prepare, execute and analyse performance tests. How long a period exists between system testing eliminating all but the trivial faults and delivery into production? We normally budget 6-8 weeks to prepare a test on a medium to high complexity environment. How much time will you have?

In E-Business projects, however, there can be some simplifying factors that make it easier to test Web applications:

- Firstly, Web applications are relatively simple from the point of view of thin clients sending messaged to servers – the scripts required to simulate user activity can be very simple so reasonably realistic tests can be constructed quickly.
- Because the HTTP calls to server-based objects and Web pages are simple, they are much less affected by functional changes that correct faults during development and system testing. Consequently, work on creation of performance test scripts can often be started earlier than traditional client/server applications.

Tests using drivers only (and not browsers) may be adequate

Normally, we subject the system under test to load using drivers that simulate real users by sending messages across the network to servers, just like normal clients would. In the internet environment, the time taken by a browser to render a Web page and present it to an end user may be a small time compared with the time taken for a system to receive the HTTP message, perform a transaction and dispatch a response to a client machine. It may be a reasonable compromise to ignore the time taken by browsers to render Web pages and present them and just use the response times measured by the performance test drivers to reflect what a user would experience.

Scripting performance test tool drivers is relatively simple. Scripting GUI oriented transactions to drive the browser interface can be much more complicated and the synchronisation between test tool and browser it not always reliable.

If you do decide to ignore the time taken by browsers themselves, be sure to discuss this with your users, technical architect and developers to ensure they understand the compromise being made. If they deem this approach unacceptable, advise them of the delay that additional GUI scripting might introduce in the project.

Which performance test architecture?

There are three options for conducting performance testing. All require automated test tools. The pros and cons of each are summarised in Table 2. To implement a realistic performance test you need the following:

- Load generation tool.
- Load generation tool host machine(s).
- High capacity network connection for remote users.
- Test environment with fully configured system, production data volumes, and security infrastructure implemented with production scale Internet connection.

5.4 Availability Testing

What is availability? Essentially, availability is measured in terms of uptime – the proportion of time that a Web-based service is available for use divided by the total time in the measurement period. In mainframe installations, uptime might be as high as 99.99% over a period of one year. That sounds incredibly high, doesn't it? 99.99% availability amounts to around 9 hours down time in the year, where a system runs 24 hours per day, 365 days per year. Assuming all systems have do be down for short periods to perform maintenance activities, this is really quite remarkable.

What might you expect of an E-commerce site? You would wish the shop to be open 24 hours per day, 365 days per year, wouldn't you? The objective of all Web sites is to be continuously available. Availability testing aims to evaluate a site's ability to stay up for a long time. Does that mean tests should span extremely long timescales? Not necessarily.

Where sites are required to be continuously available, they tend to be designs with reliable systems components, but also built-in recovery features that operate when failures occur.

These features introduce diverse routing for networks, multiple servers configured as clusters, middleware and distributed object technology that handles load balancing and rerouting of traffic in failure scenarios. Diversity and redundancy are also often the mechanisms used to allow backups, software and hardware upgrades and other maintenance activities to be performed.

Unfortunately, these configuration options are rarely well tested. Rather, they are trusted to work adequately and it is only when disaster strikes that their behaviour is

Consideration	Do It yourself	Outsourced	Performance Test Portal
Test tool license	You acquire the licenses for performance test tools	Included in the price	You rent a timeslot on their portal service
Test tool host	You provide	Included in the price	Included in the price
Internet connections	You must organise	Theirs included. You liaise with your own ISP	Theirs included. You liaise with your own ISP
Simplicity	Complicated, you do everything.	Simplest solution – the services provider do it all.	Simple infrastructure and tools solution, but you are responsible for building/running the test
Cost	Potentially very expensive	Lower tool/infrastructure costs, but you pay for the services	Low tool/infrastructure costs
Pros and cons	Could be very expensive. You acquire tools that may rarely be used in the future	Potentially cheaper, if you would have hired consultants anyway	Cheapest tool and infrastructure costs, but you still need to buy/acquire skills.
	You need to organise, through perhaps two ISPs, large network connections (not an issue for intranets)	Simpler, you need one arrangement with your own ISP only.	Simplest infrastructure solution, but you must manage/perform the tests.
	You are in control.	You can specify the tests, the services company build and execute them, you manage the supplier.	You are in control.
	Complicated – you do everything.	Simplest solution.	For majority of sites, a simple solution.

Table 2. Three Performance Architecture Compared.

understood.

Availability testing normally involves:

- Identification of the components that could fail and cause a loss of service.
- An analysis of the failure modes or scenarios that could occur where you need confidence that the recovery measure will work.
- An automated test that can be used to load the system and explore the behaviour of the system over an extended period.

- The same automated test that can be used to load the system under test and monitor the behaviour of the system under failure conditions.

Reliability (or soak) testing

These are automated tests run over an extended period, perhaps 24 or 48 hours to find (usually) obscure problems. Performance testing will flush out all of the obvious faults. The automated test does not necessarily have to be ramped up to extreme loads. (Your stress testing covers that). But you are particularly interested in the system's ability to withstand continuous running of as wide a variety of test transactions to find if there are any obscure memory leaks, locking or race conditions. As for performance testing, monitoring the resources being used by the system helps to identify problems that might take a long time to cause an actual failure. Typical symptoms are resources being used up over time and not being replaced and response times getting progressively worse over the duration of the test. You might not actually experience a failure, but given time in production, you might be able to predict what will happen.

Failover testing

Failover testing aims to explore the behaviour of the system under selected failure scenarios. In higher integrity environments, a safety analyst might conduct a Fault Tree Analysis (FTA) to identify the dependencies of a service on its underlying components. Fault tree diagrams are sometimes extremely complex with many Boolean symbols representing the dependencies on sub-systems or their components and services on their subsystems. These diagrams are used to identify the unusual, but possible combinations of component failures that the system must withstand.

Whether you use a sophisticated technique like FTA or you are able to identify the main modes of failure easily, testing follows a fairly standard process. With the automated test running, a range of failure modes are explored. It is a bottom-up approach.

- Early tests focus on individual component failures (e.g. you offline a disk, power down a server, break a network connection etc.)
- Later tests simulate more complex (and unlikely) failure scenarios (e.g. losing the power to a whole building)

You need to execute these tests with an automated load running to explore system behaviour in production situations and gain confidence in the designed-in recovery measures. In particular, you want to know:

- How does the architecture behave in failure situations?
- Do load-balancing facilities work correctly?
- Do failover capabilities absorb the load when a component fails?
- Does automatic recovery operate (i.e. do restarted systems 'catch up'?)

5.5 Security Testing

When most people think of security, they have a vision of long haired college drop-outs working into the small hours at a furious pace trying to crack a remote system. Although this image works well for the movies, it isn't helpful to our understanding the security threats that face E-business systems. Security hackers often work in teams, they are very highly automated, often they adopt a scattergun approach in acquiring targets, and they are more sophisticated than the movies usually represent them.

The intruder's attack methodology

Attackers (as opposed to Ethical Hackers, who are usually consultants) have an attack methodology. They do not attack randomly and they are extremely careful to protect themselves. Although it's amusing to think of them as sloppy burglars, tripping up and

Target acquisition and information gathering	In this phase the attacker scans the many open sources of information, large networks and individual computers. The equivalent of 'rattling the windows and doors' to find a vulnerable network or computer.
Initial access	In this phase, the attacker gains entry. Not the back door, not the front door, but through a pin sized crack.
Privilege escalation	They are in. Now how do they get root or administrator privilege?
Covering tracks and planting backdoors	Altered versions of UNIX login script, netstat, ps utilities let them return without leaving a trace. Network 'sniffers' let them see and attack all interfacing systems. Log cleaners remove all trace of their activity.

Intruder attack methodology

leaving lots of clues, this is not typical. The brief description above gives an insight to their approach.

5.6 Well documented holes have counter-measures, but are YOU up to date?

All of the major products on the market, including browsers, client and server operating system software, networking software, Web server and development products have security vulnerabilities.

If you care to read a book (e.g. reference13) or scan the Web sites dedicated to publicising security vulnerabilities, you will be staggered to learn just how many there actually are. The nature of these vulnerabilities and the tools (freely available) that exist to exploit them should make any system administrator very concerned. Of course, all of the published vulnerabilities have countermeasures or patches provided by the software suppliers, but it is the system administrators' responsibility to stay up to date and apply the patches.

Web sites such as www.cert.org (reference 23), and www.ntbugtraq.com (reference 24) exist to publicise the known vulnerabilities, risks and the availability of patches and configuration advice in all products or for a single product (e.g. Windows NT).

Security testing is normally performed in two distinct ways: security audit and ethical hacking.

Security audit

Audits are normally done in two ways. Manual security audits aim to ensure that all of the products installed on a site are secure when checked against the known vulnerabilities for those products. Essentially, this entails checking that all the security patches have been installed correctly and vulnerabilities that can be countered by configuration settings have been implemented.

Automated security scans performed by free and proprietary tools that perform ping sweeps, port scans, operating system

detection. Some (free) tools do all three at once. Using these tools, it is possible to identify what services are running on potential targets and then, to focus attention on the most exposed systems. Based on the scan results, specific countermeasures for each vulnerability can be applied.

Ethical Hacking

Using the attacker methodology, ethical hackers proceed to attempt penetration attacks. Based on initial scan results, that hackers proceed to enumerate resources, accounts, user groups and services running on the target machines using free (or their own tools). Simple as it sounds, the next task is to obtain the administrator or root password. This is done manually, using an automated tool or by listening in on login exchanges on the network. In some environments (mainly Unix), using obscure buffer overflow techniques, it is possible to log on to remote systems as root and gain access directly. Once in, the hacker can do whatever they wish on the target system.

The alternative to attacking servers through vulnerabilities in the network infrastructure is to attack applications directly. Web applications rely on the Web server dispatching HTML source code to client browsers and the source code is visible. It is a simple task to change this HTML code in a text editor, open it with the browser, and execute the transaction on the target system with changed parameters. If you see an HTML form with a hidden field called price with a value of $1999 – it couldn't be easier to edit this HTML and change price to $1.99 and then post the form to the application server. Do you think you could buy a widescreen TV online for two dollars? You might, and it might be worth a try.

Cookies are another favourite route to sensitive data. If the password for a service is generated by the system under attack and stored in a cookie, try and register several new accounts. Then examine the cookie data and see if there's a pattern. If there is and you crack the password algorithm, you can generate all the passwords on the system – past and future, perhaps. All you need now is someone else's username and you can automate as many login attempts as you like. Then you use their account to buy a widescreen TV!

Security auditing and ethical hacking require in-depth technical knowledge of the systems under attack, the tools available to assist and the imagination and persistence to crack into systems. It is a highly specialised skill and although a good system administrator or system programmer probably has the technical know-how, you really need to hire specialists to undertake this kind of testing.

6. Large Scale Integration

In section 4.3, I described the test activity associated with integration of browser pages, Web server-based objects, other server-based objects and back end legacy systems. In the context of E-business testing, Large Scale Integration (LSI) testing covers the legacy system and external system integration not already covered.

Why do we need LSI testing?

- It may not have been technically possible to perform legacy or external systems integration earlier in the project OR
- It may have been technically risky to perform LSI (there are no test systems, only production systems to test on).

An example where LSI was essential

An Internet banking application had ten major interfaces to new and legacy systems. The development of the new system itself was outsourced. The supplier, working with a customer test team were responsible for all testing up to and including system test but no testing subsequently. Integration and System testing were performed on the supplier site with all the interfaces to external systems stubbed out. LSI testing was performed by the customer in their own test environment

where interfaces to the bank's test legacy environment and other newly developed support systems could be installed. LSI required it's own separate stage because of commercial, technical and logistic constraints.

Integration knowledge

Integration testing, in principle, is a 'white box' test activity. That is, the tester needs to know something about the physical interfaces between systems. Only by knowing something about the internals, can the tester design tests that will exercise these interfaces adequately.

The tester needs to know:

- The details of the internals of the interface.
- The nature of the system-system dialogs.
- How to exercise the interface from the application user interface.
- How to create test data to exercise the interface.
- How to find evidence that the interface works.

One of the problems of LSI testing is that it can be difficult to find details of interfaces. It's not just the interface details that cause problems. It may be that there is no documentation available at all for the legacy systems. In our experience, the LSI test team often has to write the document describing a system's interfaces as well as the test plan to ensure they work.

We'll assume interface documentation exists. Typically, to define the integration test plan the tester follows the same general process:

For each interface, identify the dialogs between systems and which business or system event triggers them to work.

Derive test cases for success and failure to negotiate each step in the dialog.

Derive test cases from the interface data validation/use descriptions to ensure valid data is transmitted, invalid data is rejected and that the storage and use of data in each interfacing system reconciles.

Define your test environment/infrastructure needs early so you get them in good time.

Conducting integration tests

Integration tests tend to be either very simple and easily automated or complicated and have to be manually executed. In general, early tests focus on the correctness of the interface calls. Later tests (usually automated) focus on memory leaks, loss of synchronisation between systems and failure and recovery of client, server or network.

6.1 End-to-End Functionality

How does end to end testing differ from Large Scale Integration testing? Broadly, End-to-End tests are usually associated with user acceptance. Assuming that the technical testers have proven the interfaces between the new system and other systems work, the imperative for a business wishing to deploy the new system is to ensure the system supports the intended business activity. For example, if a system supports on-line purchase and delivery of books:

- Can the user search for a book, add it to a shopping basket and place an order?
- Can the customer credit card be validated, payment authorised and processed successfully?
- Does the legacy order processing system receive the on-line order accurately?
- Is the book in stock, located in the warehouse, packed, labelled and dispatched correctly?
- Are "order confirmation", "progress notification" and "thank-you for ordering" emails sent at the right time and reaching the customer reliably?
- And so on…

Ultimately, the sponsors of the system want to know whether the new system meets the

cardinal business objectives of the project. To demonstrate this, testers must trace paths through the business process and develop scenarios that will exercise the system and provide evidence that the system supports the business process.

Compared with system testing, it may take a relatively small number of test cases to give confidence that the system works correctly. The difficulty in staging such tests at all, is they require the entire technical architecture.

7. Post-Deployment Monitoring

Imagine that you were investigating an E-commerce Web site and thinking about using it for the first time and it failed. Would you report a problem to the company running the site? It's not likely that you would. Once again, the general attitude to Web sites is just like retail stores. If a store is closed, service is slow or difficult to use, you just walk on by and visit the next store in the street. It's the same with Web sites. Search engines and portals offer you a multitude of websites matching your search criteria so it is easy to click on a different link. With some E-Business, for example electronic banking, this is less likely. After all, the bank has YOUR money and you want it, so you would complain pretty quickly.

Unlike users of our internal systems, we cannot rely on users of our Web site to report problems.

With this background, it is sensible to monitor your site in production. Is this testing? It's not the traditional role of testers to test a site in production, but monitoring a live site involves exactly the same activity as testing a development site.

There are four main areas where post-deployment monitoring is appropriate:

- **Availability**
  - Is our site available - NOW?
  - You might check this every 15 minutes, every single day.
- **Link checking**
  - Does your site rely on many links to external sites?
  - Have target sites and resources been moved?
  - You might check this once per day.
- **Object load times**
  - Are our pages (and images) getting bigger over time?
  - Is the database becoming huge?
  - Are key transactions getting slower?
  - You might check this once a week.
- **Security**
  - Are our security policies still sound?
  - Are we vulnerable to a new form of attack?
  - Most likely to be done by internal or external consultants using scanning tools.
  - You might check this every time a new vulnerability is announced.

The report displayed next page was produced for the Evolutif Web site by the NetMechanic tool. This is typical of the reports that these test tools produce.

What can be checked?

Test sites can 'Ping' your site to see if the server hardware is responding.

HTTP requests can tell whether the Web service is running.

HTTP GETs of selected Web pages can tell whether pages are downloadable and the speed can be monitored over time.

Calls to CGI programs can execute selected test transactions to ensure that connections to back end systems are also operating properly.

Link checkers scan all pages for links, follow them until they result in off site pages, they

Report For: www.evolutif.co.uk
Date: Thu 28 Sep
Time: 6:00 - 13:00 USA Eastern Time
Server Type: Microsoft-IIS/4.0

Overall Rating: Fair

Performance Summary			
Event	Your Server's Average	NetMechanic Average	Percentile
Host Ping	*	277.26 millisec	th
DNS Look Up	0.18 sec	0.13 sec	9 th
Connect Time	0.09 sec	1.11 sec	62 th
Download Time (10k file)	0.50 sec	1.10 sec	37 th
Timeouts	0	-	

Host Ping:	Indicates the network speed between your server and ours.
DNS Look Up:	Indicates the time required to look up your server's address in the DNS database.
Connect Time:	The time required to open a network connection to your server.
Download Time (10k file):	The time which would be required to download a 10k file from your server.
Number of Timeouts:	The number of times your server failed to respond within 30 seconds.

Warning areas are highlighted in dark yellow; problem areas are highlighted in red.

are checked and the test stops.

Security scanners can evaluate exposure to product vulnerabilities.

There are three options for post-deployment monitoring of Web sites:

Do it yourself

In principle, you could reuse the automated tests that you prepared during the development phase and execute these on a remote machine on a different site to where your systems operate. You would also want to base this test machine on a site connected through a different ISP to ensure ISP problems don’t affect the test machine as well.

Remote monitoring services

There are now many companies offering continuous site monitoring. These are often the same companies that have remote HTML validation and speed-checking portals. In these cases, prices can be a low as a few dollars per month to monitor a site or specific pages. The companies may have remote agents-based in many places to provide comparative timings. The more sophisticated services are more expensive, of course. Most offering include regular (e.g. 15 minute) checks and alerts in the form of email, fax and pager messages. Weekly and monthly trend reports may also be offered as part of the service.

Consultants

Consultants are most often used for security audits. The general recommendation is that you should do a security audit every three months, although it might be most wise to do spot checks when every new vulnerability (on products you use) is publicised to ensure you are not exposed. Be aware, the hackers monitor the same security alert email lists as you do. When a new alert is published, they have a window of opportunity to exploit it before the patch is installed in all exposed sites.

References

E-Business Testing

1. http://www.evolutif.co.uk/ E-Business Risk-Based Testing, Part 1: E-business Risk and Test Strategy. Paul Gerrard, June 2000.

Tools and tool Listings

2. www.softwareqatest.com/qaweb1.html Comprehensive Web (and other) test tools listing.

3. http://Validator.w3.org Link checker.

4. http://www.netmechanic.com Link checker and other site validation.

5. http://www.htmlvalidator.com/ CSE HTML Validator it user configurable HTML, XHTML, and WML syntax checker available for Windows 95/98/2000/NT. With a built in browser facility (needs IE4 or later), you can browse pages, see the page source code and validate it at the same time.

6. http://www.cast.org/bobby/ Bobby is a free application that will analyse Web pages for their accessibility to people with disabilities. It will also find HTML compatibility problems that prevent pages from displaying correctly on different Web browsers. Available as a Web-based services and downloadable tool.

Testing Papers

7. http://www.evolutif.co.uk/ Client/Server Performance Testing, Paul Gerrard and Andy O'Brien 1995. This paper describes the end-to-end process of planning. Preparing, executing and analysing performance tests in a client/server environment. Although it doesn't mention the Internet, the principles appear to be unchanged since the paper was written in 1995.

8. http://www.evolutif.co.uk/ Testing GUI Applications, Paul Gerrard, 1997.

Books

9. How the Internet Works, Preston Galla, ISBN 0-7897-2132-5. Best selling introductory text.

10. Webmaster in a nutshell, (five books on CD-Rom), O'Reilly

- HTML: the definitive guide
- JavaScript: the definitive guide
- CGI programming for the world wide Web
- Programming Perl
- Webmaster in a nutshell

11. Understanding Electronic Commerce, David Kosiur, Microsoft Press

12. Designing Web Usability, Jacob Nielsen, ISBN 1-56205-810-X. A very readable, well illustrated and up to date book on Web usability.

13. "Hacking Exposed: Network Security Secrets and Solutions" - McClure, Scambray and Kurtz ISBN 0 07 212127 0 http://www.hackingexposed.com

Web Resources

14. E-commerce Awareness Forum newsletter, www.mbs-program.com

15. Web Accessibility Initiative: http://www.w3.org/WAI/www.webreference.com/authoring/design/tutorials.html

16. General Web usability. This is Jakob Nielsen's Web site focusing entirely on usability issues: http://www.useit.com/

17 Response times: the three important limits: http://www.useit.com/papers/responsetime.html

18. How to conduct a heuristic evaluation: http://www.useit.com/papers/heuristic/heuristic_evaluation.html

19. Software Usability Measurement Inventory [SUMI] & Website Analysis & Measurement Inventory [WAMMI]: www.ucc.ie/hfrg/ & www.acm.org/~perlman

20. Software for Use, Constantine and Lockwood, ISBN 0-201-92478-1. Comprehensive and up to date treatment of usage-centred design with information on all main methods of usability test and evaluation. See accompanying Web site: http://www.foruse.com.

Security

21. Internet Security Policy: A Technical Guide, NIST, http://csrc.nist.gov/isptg/html/ISPTG-1.html

22. eCommerce Trust Study, Cheskin research, http://www.studioarchetype.com/cheskin/

23. http://www.cert.org - generic security bug watch run by the Software Engineering Institute – register and get regular advisories on new vulnerabilities in the major OS, Web and browser technologies.

24. http://www.ntbugtraq.com - same as the CERT site above, but for Microsoft NT only.
25. Firewalls Frequently Asked Questions – excellent starting point for this topic, http://www.faqs.org/faqs/firewalls-faq/

Other References

26. http://www.w3.org/TR/html4/ HTML 4.01 Specification, W3C Recommendation 24 December 1999

27. World Wide Web consortium: http://www.w3.org.

Software Configuration Management for the Web

Michael K. Jones, jonesaz@uswest.net
Director of Quality Assurance, TransactPlus, Inc.
Assistant Professor, Western International University, Phoenix, Arizona, USA

Software configuration management must be practiced for any software system from the moment any work on the software system begins to when support ceases for that system. In the E-World, which is an expression used to describe the environment in which Web applications are developed and utilized, effective and time-efficient software configuration management must be practiced such that baselines are identified and changes are tracked to those baselines. However, this software configuration management effort must not impede development and must be clearly understood from a functional basis.

The overwhelming time to market requirement for E-World development efforts can not tolerate any impediments to development time and responding to client needs. This high emphasis on time efficient processes has not been true for many software development projects in other environments in the past as a rule, and pragmatism has surely not governed many efforts that had to observe numerous government regulations.

The theoretical model that is most commonly used in reference to software configuration management is one that has a typology of configuration identification, change control, status accounting, and configuration audit. This typology is not functionally oriented and is consequently either misapplied with gaps in coverage or impedes the development of the software system by being oppressive and bureaucratic. This model was developed with hardware in mind as its primary target and consequently must be interpreted for software projects.

A better model for software configuration management that is clearly understood and is scaleable is the subject of this paper. Software configuration management can be functionally broken out into the areas of 1) version control, 2) document control, 3) change management 4) build management, and 5) release control.

By discussing each of these functional areas of software configuration management at length, a realistic and doable model of software configuration management will be seen from the explanations and examples supplied in the paper. The reader of this paper will then, hopefully, leave with a better appreciation of what to expect in software configuration management efforts in the future, both as a developer and as a client.

Introduction

Software configuration management is a crucial activity for any software development effort. The software configuration management activity, however, must not delay or impede the rapid software development schedule necessary to meet the harsh time to market needs of the E-World.

Consequently, effective and time-efficient software configuration must be practiced with all efforts having a justification in functional value. The software configuration management theoretical model that is most commonly referred to in literature does not easily correspond to the functions that must be accomplished through software configuration management activities. A better model that is functional in derivation, and that will be clearly understood and will be easily scaleable is the subject of this paper.

A functional model of software configuration

management is organized into the areas of 1) version control, 2) document control, 3) change management, 4) build management, and 5) release control. This typology corresponds directly to the functional tasks that must be performed for a project and also agrees with the typology of the major software configuration management tool vendors. Each area will be defined and the practices or techniques that must be implemented for Web applications will be discussed.

The Theoretical Model

The theoretical model of software configuration management arose out of post World War II efforts in the aerospace industry for the United States Defense Department and was a spin-off of previous successful efforts for hardware configuration management according to Berlack (1992). This model has continued to be referred to by textbooks on configuration management, Buckley (1996), and by textbooks on software engineering, Peters (2000). The model consists of four elements: configuration identification, change control, status accounting, and configuration audit. Although this model still stands head and shoulders above the Microsoft model of the "daily build" as discussed by Pfleeger (1998), the traditional model still remains encumbered by a focus on bureaucracy instead of project functionality.

The Microsoft daily build technique, for those who are unfamiliar with it, consists of declaring the baseline by whatever was "thrown over the fence" to the build manager at either the beginning of the day or the end of the day, depending on the project. Why this technique is not held in high regard by at least some in the software engineering community, is that builds are not a result of conscious decisions about what functionality should be included in a baseline, but only a result of frenetic programmer activity that is personality dependent.

Definitions for the traditional model as provided by Buckley (1996) provide some insight into this conclusion:

Configuration Identification – "Configuration Identification includes the selection of configuration items; the determination of the types of configuration documentation required for each configuration item; the issuance of numbers and other identifiers affixed to the configuration items and to the technical documentation that defines the configuration items configuration, including internal and external interfaces; the release of configuration items and their associated configuration documentation; and the establishment of configuration baselines for configuration items"(p.259).

Change Control – "The systematic proposal, justification, evaluation, coordination, approval or disapproval of proposed changes and the implementation of all approved changes in the configuration of a configuration item after formal establishment of a baseline"(p.259).

Status Accounting – "The recording and reporting of information needed to manage configuration items effectively, including:

a) A record of the approved configuration documentation and identification numbers.

b) The status of proposed changes, deviations, and waivers to the configuration.

c) The implementation status of approved changes.

d) The configuration of all units of the configuration item in the operational inventory (p.260).

Configuration Audit – "The verification of a configuration item's conformance to specifications, drawings, and other contract requirements (p.258).

From these definitions, there is a strong inference, substantiated by the author's personal experiences on past projects that

used the traditional model (dictated by DOD or NASA standards), that extraneous paper documentation becomes the overriding "end" as opposed to the "means". Along with this paper, came a bloated personnel count with "checkers" of the "checkers". Hence, large amounts of extraneous paper documentation and superfluous personnel equate to needless bureaucracy.

The Functional Model

In contrast, the functional model that this paper advocates is strictly based upon the correspondence of software configuration tasks to the tasks found in the basic software development model. The basic software development model, according to Sommerville (1996), who is the author of one of the most commonly used textbooks on software engineering and consequently one that most people might be familiar with, consists of four fundamental activities which are common to all software development processes: software specification, software development, software validation, and software evolution. The functional model of configuration management maps the functions of version control, documentation control, change management, build management, and release control to the development model.

The model is called a "functional" model because the typology is based on tasks that are commonly called out on a WBS (Work Breakdown Structure) and there is no need for interpretation of task versus configuration management area type. A functional emphasis is important for the E-World, i.e. Web applications, because all actions must be as time-efficient as possible to meet deadlines and yet control of baselines and changes must still occur. There is not the luxury of pondering over what configuration management means according to some abstract model, but rather a driving concern with getting tangible tasks performed quickly. A "functional" model is focused on getting those tasks done, not with generating paper.

Mapping the functional software configuration model to the development model, version control takes place at the conclusion of development with formal software baselines prior to validation. Documentation control takes place at the conclusion of specification and then continues throughout with traceability of documents to software baselines. Change management is initiated immediately following the first instance of the use of version control or document control. Build management occurs with the initial repeatable documentation of how to construct the first formal baseline with updates then being a constant necessity. And release control is performed at the conclusion of validation such that all versions of the system that are released to outside parties are approved, recorded, and tracked against requests for defect resolution and enhancements. A description of typical tasks and activities for each functional area follows and is organized on the basis of "What", "Why, "When", "Where", "Who", and "How" for the purposes of clarity and definition. In addition, the special needs and consideration of Web applications will be discussed.

Version Control

What: Version control combines procedures and tools to manage different versions of configuration objects that are created in the software development process, according to Pressman (1997). Configuration objects are identified, for Internet applications, by completing a physical architecture diagram during the design process. It is only at that point that objects may be discerned in a configurable form. The physical architecture diagram had to be preceded by the creation of the domain model and the object model for successful design derivation.

Why: Version control provides for visibility, control, traceability, and monitoring of software components and systems, and of documentation. It is a necessary function for

successful project management and the implementation of a "planned" system architecture.

When: Version control is instituted for software systems after development is complete, so that baselined units can be employed in system integration and validation testing. Version control is then repeated with new versions for those items as changes are introduced to those baselined items.

Where: Version control is performed through version control utility software located on a system that is accessible by all participants on project team (i.e. developers, testers, project managers, and quality assurance.

Who: Version control is performed by multiple parties with different roles. Unit developers check units in and out for their work. System developers construct and identify specified groups of units as configured systems. Testers check systems out for testing. Project managers approve these configurations at project milestones. All of these activities are performed by project personnel with the appropriate privileges granted in the version control system. The administrator of the system, however, should be a configuration management analyst who is responsible for the administration of version control throughout the organization, in addition to other areas of responsibility such as change management and release control.

How: Version control is accomplished mainly by software tools with some set of the following capabilities:

- Version any type of file
- Forward/reverse deltas
- File compression
- Branching
- Automatic branch creation in certain situations
- Version labels
- Visual differences
- Visual merging
- Parallel development
- Support projects as well as files
- Support sub-projects
- Unlimited directory hierarchy
- Recursive commands through hierarchy
- Track changes to project structure
- Project/file sharing between multiple applications

In the preceding nomenclature, a definitive description of version control for the functional model was presented. This description provides the foundation for a discussion of special E-World needs and considerations.

For Web applications, tools are a necessity as they greatly increase efficiency over manual methods with hard media and much personnel. The drawback to using automated tools is the cost of sufficient licenses to support all the users of the configuration management process.

Other considerations for Web applications are the necessity to baseline not just source code, but also executable code, graphic images (GIF and JPEG files), data files, and any type of digital article necessary to create the finished system. In many cases today, content management is simply a new name for version control of Web applications in production, but the tool utilized has been optimized for control and delivery of controlled items from the repository to the production server.

One special concern, due to Web applications, is the need to check for the legal right to use and archive the images received for version control. With very creative and eccentric designers, copyright authorization may have been neglected. It is recommended that metadata of copyright permissions be captured with any images at

the same time that they are accepted into version control. Additionally, all HTML code should have a copyright protection statement inserted so any unauthorized use can be prohibited if Internet users cut and paste the code.

Document Control

What: Documentation control combines procedures and tools to manage and preserve versions of all identified documents in the development process model.

Why: Documentation control provides a means to preserve and retrieve documents used in the development model so that traceability to software baselines for decision-making is maintained and a basis for litigation and negotiation is available.

When: Documentation control is performed whenever a document in the development model is approved by the required parties. The master of the document is archived in a controlled area and a copy is used in daily work.

Where: Documentation control is performed through documentation control utility software located on a system that is accessible by all participants. Signed hard-copy masters are kept in a controlled area.

Who: Documentation control is performed by all individuals who create documents throughout the development process. After the designated approval of their documents, they are responsible for seeing that those documents are placed in documentation control. The administration of the documentation control function should be the responsibility of the project configuration management analyst.

How: Documentation control is accomplished mainly by software tools with the following capabilities:

- Forward/reverse deltas
- File compression
- Version any type of file
- Branching
- Automatic branch creation in certain situations
- Version labels
- Visual differences
- Visual merging
- Parallel development
- Support projects as well as files
- Support multiple threading
- Unlimited directory hierarchy
- Recursive commands through hierarchy
- Track changes through business cycle
- File sharing between multiple applications

Documentation control is broken out separately in the functional model, due to tool and organization issues. In the realm of tools, it is certainly possible to use most of the version control tools on the market for document control, if they will accept the file type that the document was written in. However, some document control utilities are sold with server licenses strictly and not by user licenses, so that they may make more sense in providing a larger group of users access and/or supporting in a cost-efficient manner access to a corporate knowledge management repository. Version control tools, in contrast, are usually sold by user licenses and may be tied to other tools that the document users may not need.

One special consideration for E-World applications is that the prospective document control system must have the capability to keep hyperlinks embedded in the documents. Other issues may be globalization concerns that revolve around the use of non-Western alphabets in Web pages and documents, and the consequent need to have the ability to utilize Unicode, a new font mapping standard.

Change Management

What: Change management is the systematic proposal, justification, evaluation, coordination, approval or disapproval, and implementation of all approved changes to the configuration of a system after the initial baseline of that system.

Why: As uncontrolled change leads to chaos, controlled change enables changes to the system to be accommodated efficiently and effectively both in time and cost, and permits a planned resolution of reported defects and requested enhancements for that system.

When: Change management may be initiated at any point in the project life cycle after the first baseline, through the creation of some form of a change management request that is submitted to a configuration control board for resolution. Changes are always targeted against specific baselines, whether those baselines are versions of code, versions of design, or versions of requirements. It is very possible that a requested change to a code version will have a backward effect on both design and requirements. Changes include both reported defects and requested enhancements.

Where: Change management is accomplished usually through the use of two configuration management tools: 1) a change control tracking system, and 2) a version control system. The change control system must be linked to the version control system. The change control system registers the initial appearance of the change request and routes the request to analysis review, after which the request awaits the decision of the configuration control board as to disposition. The system then routes the request to the designated implementer, records a description of the implementation, and routes the request back to the board for approval of implementation and incorporation into a release baseline. All changes must have links to specific items in the configurations captured in the version control system.

Who: Change management involves the entire project team and clients. A change management request may be created by any member of the team or by any client. The configuration control board is usually made up of the project manager, a technical lead, a configuration management analyst for support and administration, and quality assurance. The board determines the resolution of all requests and defines all releases.

How: A successful change management system is attained through cost-effective and time-efficient processes, trained and professional personnel, and well-chosen configuration management tools. A configuration management system should not result in a burdensome bureaucracy, but in a documentation of decision paths that show that when changes were made to the system, the changes were analyzed before implementation and the implementation itself was reviewed prior to incorporation in the client load. All changes link directly to items in a controlled version system, and conversely all ensuing versions after the initial version have links to change management requests. Those links to the version control system are both links to the previous version where the defect was found or the enhancement needed, and to the new version where the change was implemented.

For Web applications, with time always of the essence, a time-intensive change management system based on paper and numerous meetings is not probable and in many cases possible. Successful change management in the E-World hinges around two things. The first factor for successful change management is automating the change process using a tool with automatic notifications and forwarding once system approved decision-makers have made a choice of actions. The second factor is empowering the Project Manager (or Product Manager depending on whether the project is a custom application or a product) to act as a one-person change control board. It is up to the Project Manager to solicit and obtain the

views of others towards the Project Manager's approval of changes.

Build Management

What: Build Management is the function that either performs or verifies the build of all configured baselines through controlled documentation.

Why: Build management provides for traceability and repeatability for all configured baselines such that deliverables to clients are traceable to source articles.

When: Build management is performed when any configured baseline is produced.

Where: Build management is performed at any location whenever an original configured baseline is produced.

Who: Build management may be performed by any designated project personnel, but the record of how that build was accomplished, as well as a documented process that details the technical steps to perform the build, must be reviewed and archived by the configuration management analyst.

How: Build management may be accomplished by either through an automated utility that interacts with a version control tool, or through a recorded script that captures the commands and references necessary to repeat the build with no differences found in the products of builds performed separately. The script, upon version control of the baseline, would also be placed in an archive and linked to that baseline.

Build management for traditional software configuration management individuals is an unfamiliar subject, but it is a major challenge to be dealt with for Web applications. The reason for this is the complexity of Web applications due to the interdependencies of varied software technologies and the hardware environments they operate within. For the theoretical model a simple version description document was sufficient for a later configuration audit.

For Web applications, not only must all software and hardware be listed in sufficient detail to be clearly identified, but also even the sequence of how that hardware and software is set up and installed must be recorded if the successful building of a production server is to take place. For Web applications that the author has been responsible for, a document defined as a "Configuration Specification" was created that records the identification of all hardware items, the identification of all software items, the setting of any switches and breakers, all commands entered into the system, and the proper sequence of the preceding actions. This documentation is far beyond what is usually found in a Version Description Document. Furthermore, this document is validated by being utilized to create from scratch the test servers and then finally the production servers.

Release Control

What: Release control is the function that collects, documents, and transmits all deliverables to points external to the company. A configuration audit of all deliverables must be performed to ensure an accurate record of configurations and validated traceability to build procedures and system documentation. (It must be noted here that the term configuration audit is misunderstood by many new to software configuration management as a process audit. It is not a process audit, such as would be performed by Quality Assurance auditors, but a product audit against the software to be prospectively delivered to outside parties.) For Web applications, the configuration audit must be conducted against the "Configuration Specification." The audit is the validation of the "Configuration Specification," and is the build of a production server from scratch. This is conducted in a controlled environment, usually a staging or test lab.

Why: Release control ensures that traceability as to configuration, authorization, and destination for all deliverables is kept and auditable. Release control records are a starting point when answering client questions as to what configuration the client currently has when changes are discussed.

When: Release control is performed in the life cycle of custom code and product code as the delivery interface to the client. When changes to deliverables have to be implemented during installation, those changes must be reported to Release Control so that a true and accurate record of client configurations is kept.

Where: Release control is performed at the company, prior to delivery to clients. When changes occur at the client site during installation, those changes are required to be reported back to Release Control for accurate record keeping. From release control, all changes are then incorporated into the version control and change management system. Approval is implicit due to the involvement of the project management team during client installation. An important point to make is that these five functions all support and interact with each other.

Who: Release control is performed by a configuration management analyst who supports multiple projects for their release of deliverables to customers. The analyst performs this function, in addition to also providing administration of version control and change management systems. In the case of emergency transmissions due to the immediate needs of the client, when Release Control is unavailable, project managers are always empowered to deliver the appropriate fix to the clients. The next working day, however, copies of that transmission and the proper records must be given to Release Control so that the organization has records of that delivery.

How: Release control is accomplished through the use of the version control utility by keeping a record of the configuration of all external deliverables. The configuration management analyst should also check for agreement of appropriate personnel as to whether the deliverables should be transmitted to the outside parties. Also, the analyst should perform a configuration audit, if one has not been performed prior to this point, to validate the agreement of software to the system documentation. This would be the "Configuration Specification" for Internet applications. The audit will assure that all items of the delivery are in agreement and the traceability of the software and hardware is to the correct build procedure, i.e. the "Configuration Specification" for Internet applications. Finally, hard copies of all release approval signatures and dates should be kept for audits and possible future litigation.

Release control is identified as a specific activity under the functional model due to its critical nature in the E-World. All changes to a production Web application must be tracked and have traceability to controlled items in a repository, so that effective content management can occur. Due to the need to expedite defect removal and in some cases accelerate enhancements to site functionality, the project manager (or product manager) is always empowered to make decisions for releases without involving others.

Migration Structure

For any software configuration management model and to optimize the functional configuration model, a compartmentalized physical facilities structure for the controlled migration and implementation of software systems has been found to be effective, particularly so for the E-World. This migration structure for Web applications consists of a defined development environment with control of the project configuration decided by the project manager, a staging environment for the testing of the developed systems with formal controls in place for changes to the system

under test, and a production environment where no changes are permitted as the system operates in the critical 24 by 7 by 52 time frame of reliability and non-down time mode. As a software system moves from development to staging, entrance criteria must be met as to the completion of required documentation, a defined baseline, and the preparation of test procedures. Similarly, as a software system moves from staging to production, entrance criteria must be met as to complete system documentation, and completed test documentation that provides a basis for risk management as the reliability of the software system in the production environment.

With the total revenue stream of pure play (all business taking place on the Web) e-commerce businesses dependent on their Web sites, a comprehensive risk reduction strategy for downtime of the Web site is a prime necessity. Consequently, hosting operators are now usually contractually liable for downtime as a result. All of this is driving more Web application development organizations to a controlled migration structure for their projects.

System Automation

For the successful utilization of the functional model of software configuration management, most especially in the E-World, automation of the software configuration processes is essential. Providentially, numerous software tool vendors have met this need with suites of software configuration management tools. These tools are organized along the functional model and not the theoretical model, with specific tools in each suite focused respectively on version control, documentation control (in some cases the same tool as the version control tool), change management, build management, and release control. There is no suite or individual tool that corresponds to the typology found in the theoretical model. In fact, the functional typology is what the vendors address in their specifications not the theoretical model. A prime source for information about these tools is the "Configuration Management Yellow Pages" as found on the World Wide Web with the URL of http://www.cmtoday.com/yp/configuration_management.html. Automation of the software configuration management functions provides, among other things, for the automatic requirement of approved change requests for every baseline once the initial software baseline has been achieved, the mapping of baselines to change requests at release control reviews, the linking of build procedures to every baseline, and the automatic traceability of system documentation to software baselines.

For Web applications, automated software configuration management accelerates process decisions. If the decision was between a time expensive manual system and not doing configuration management, it is possible that the latter choice might be taken, even with a consequent loss of control and the lack of change records. The use of automation makes sure that the possible decision to not do configuration management is never considered.

System Interfaces

Having discussed system automation above, it is necessary to discuss the need for several required system interfaces for a software configuration management system in an E-World environment. All of these interfaces must be either integral with the system or an API must be written for back and forth transmission of information. First, by linking the change management utility with the organization's e-mail system, automatic notification of required actions or decision can be made and recorded, with the change request then being forwarded on to the next decision point for further action. Second, due to the need for the smooth and expeditious reception and response to client or customer requests for defect resolution or enhancement, an API is necessary between the customer support automated software and the software configuration management

tools. By this means, a customer request may be received and then transferred to the software configuration management system, with real-time status available to the customer support representative at any time for the decisions or progress made on requests. Third, the content management delivery software system must be linked to the software configuration management system with an API for the necessary adjustments of the site software in the E-World that are necessary to support new data and image needs that must be displayed and manipulated. Fourth, the system site software must be controlled through the software configuration system to avoid conflicts between operating systems and third party software that have been purchased to operate with the Web sites.

Conclusion

This paper has tried to discuss the recognition of a new standard model for software configuration management and how that model makes possible pragmatic software configuration management in the E-World. The paper has described the old theoretical model of software configuration management and defined the new functional model as version control, documentation control, change management, build management, and release control. Individual considerations for Web applications have been reviewed for each area of the functional model. The paper has discussed each of these areas in detail through using a "What", "Why", "When", "Where", "Who", and "How" typology to provide clarity and definition. Ramifications of the functional model of software configuration management have been discussed as regards to migration structure, system automation, and system interfaces that are necessary for success in the E-World. In particular, it has been noted that the functional model of software configuration management matches the organization of the software configuration management tool suites currently available for software development.

In conclusion, the paramount principle for the guidance of performing software configuration management in the E-World should be one of pragmatism. In the past, pragmatism has not been the paramount principle in software configuration management for many projects as exhibited by their paper and personnel bloat. Definition and communication should be the ultimate goals for any software configuration management activity and not barricades and forms. Through taking a functional view of the tasks of software configuration management, efficiency can be improved by seeing rational reasons related to functions for the tasks to be performed. Furthermore, scalability can be matched to that of the development model more congruently. The E-World will tolerate no paper for paper's sake, and so consequently a functional view of software configuration management will better ensure that the work of software configuration management always brings value.

References

Berlack, H. (1992) Software Configuration Management. New York: John Wiley & Sons, Inc.

Buckley, F. (1996) Implementing Configuration Management: Hardware, Software, and Firmware. Los Alamitos, California: IEEE Computer Society Press.

Peters, J. & Pedrycz, W. Software Engineering: An Engineering Approach. New York: John Wiley & Sons, Inc.

Pfleeger, S. (1998) Software Engineering: Theory and Practice. Upper Saddle River, New Jersey: Prentice Hall, Inc.

Pressman, R. (1997) Software Engineering: A Practitioner's Approach. New York: McGraw-Hill.

Sommerville, I. (1996) Software Engineering. Harlow, England: Addison-Wesley Longman Limited.

Companies

Cooling down... laying off!

The Internet miracle is now showing a new trick: making jobs disappear at e-consulting companies. Recently MarchFirst said that it would lay off 1000 employees or around 10% of its workforce to save $ 100 million. Razorfish plans to cut with a similar percentage in its employees' ranks. Deep troubles also for iXL that will let go 35% of staff (850 employees) and close seven offices. Finally, Scient will lay off 25% of its work force (460 jobs) and close two offices.

The cause is not that development of Web-based applications is disappearing, although the failure of "dot.com" companies has hurt demand, but these companies are also spending a lot of dollars in infrastructure items like fancy offices or salesmen. The flow of consulting fees has to be big enough to cover these costs, which is not currently the case as competition is big in this area. According to Scient CEO, his company was undercut by as much as 40% on some projects by Big Five competitors. Traditional companies are also taking more time to examine their Internets needs.

Rational buys Catapulse

Rational has completed in November the acquisition of Catapulse, a company founded in October 1999 by two Rational's co-founders Paul Levy and Michael Devlin. Rational already owned 35% of Catapulse. This company is specialized in leasing software development tools over the Internet in cooperation with firms like IBM and Loudcloud.

This acquisition is the confirmation that the current trend of application service providers (ASP) over the Net is also spreading in the software development industry.

In Others' Words

Light structuring

"How do you structure a process as complex as software development without killing it? Without creating that deadly momentum that translates good intentions into a process straitjacket? It is perhaps the fundamental problem of our profession. [...]

In the time since my fateful experience with methodology, I've done a lot of development and visited a lot of development organizations. One thing that's new in the intervening years is the proliferation of companies that sell software commercially and the resulting size of that industry. Software development's mainstream is no longer large companies building systems internally, but smaller companies building systems for sale to other companies and consumers. Not surprisingly, the fierce

crucible of market competition has brought changes in the supporting product-development process. And guess what? The vision we had - of a calm, rational development process, a kind of engineering - it's toast. Gone. [...] I don't think I've visited any commercial software companies where the development process is truly calm and rational. [...]

Here are some characteristics of modern software development processes that I'd suggest are pretty representative of strong development organizations.

- Superstar individuals are the difference between success and failure. Cruise the code and you'll always find disproportionate evidence of a few strong developers - way disproportionate. Said another way, this software thing is a craft, and stellar craftspeople are the key. It's one of the features we complained most about in our methodology days. We used to talks about taking the art out and making the process more scientific. That's looking increasingly like a fantasy we indulged in from within protected internal IT enclaves.
- Requirements change continuously. We used to talk about nailing down the requirements up front, engaging users in contract they would sign off on upon delivery. In market-facing organizations, there is little time for this. It's a fact of business life that important stuff comes up later, and a process that deflects late changes risk irrelevancy.
- Lots of things happen in parallel. We used to argue that an error caught in testing was a lot more expensive that an error caught much earlier in, say, requirements definitions. In the massively parallel development processes of many new organizations, it's challenging to say what earlier means, except in general sense. The best development organizations have worked on making all changes less costly, and on making development a set of concurrent rather than sequential steps. The build cycle organizes development. In a sense, the build cycle has collapsed the stages of our old methodology into a tightly compressed timeframe - sometimes a day (or less). What we're doing is not sequential with support for iteration, like my old version of methodology; rather it's fundamentally iterative.

One very successful company I know deliberately eschews formal process and even organizational structure. They hire the smartest developers they can find, organize them into very small teams, and then get out of the way, holding individual developers responsible for the quality of their products (as determined by automated testing routines). Rather than assemble requirements documents, they use supertalented developers to build overnight prototypes. Even if the prototype is miles off from what the customers want, the resulting customer reaction is far more informative than most conscientiously constructed document, and the developer learns what's needed a whole more quickly. The development VP there once wrote operating systems at a big company and knows the shortcomings of methodology-intensive ways.

These New Age companies structure their processes only to the extent that it works. If something doesn't work, they either stop doing it or lose in the marketplace. What they evolve in the way of process does sometimes seem to fit with more traditional notions of software methodology. Rapid prototyping is not particularly new. And these companies do adopt more structure as they grow, because they are to some extent overcome by the growing complexity for informal processes (at some point, coordination between different developers gets really hard.) In addition, a lot of what they are doing is simple experimentation, some of which will be rejected eventually. But when it comes to structure and the everpresent need to get new product to market, it's a balancing act - a tension that must be managed. It's not about finding an optimal process that can be designed

perfectly enough to solve the problem completely.

In this world, the real phantom menace is the seductive idea that you can construct an optimal, engineeringlike process for developing high-quality software products. It's this idea that gets the ball rolling toward the stultifying "filling in" to which my treasured notion of methodological excellence succumbed so long ago. It's what threatens to steal the vitality from agile development organizations, to turn them into stodgy corporate IT departments clones. A big part of the problem is with our analogies - the patterns to which we aspire. Is software development really like manufacturing? Is it really like engineering? Maybe we need some new patterns. [...]

What then should we do? [...] One of the best things we can do is try to learn how other young, market-facing, rapidly growing companies are doing things. Often these companies are not encumbered by standard patterns and language, simply because they don't have a history of using them. Many are doing a lot of experimenting, which makes it hard to separate what they'll keep doing from what they're just trying and will eventually stop. Also, some of what they're doing might not be relevant to you because of differences in your situation. But plenty nuggets of goodness are there to be had."

Source: Robert D. Austin, "The Phantom Menace", IEEE Software, September-November 1999.

This article pose important questions on why and how to implement process-like attitudes in a software development organization. But if all developers and users were geniuses, there will be no problems in our profession. Oops, I forgot managers! The question is how to insert in software product some common and long-term values when the many of the software "craftspeople" are individual and short-term oriented. The requested qualities of software project results are not similar one year after its completion.

Coming next in Methods & Tools

- Database Design with UML
- Introducing QA in a Web start-up

Classified Advertisement

METHODS & TOOLS is published by **Martinig & Associates,** Rue des Marronniers 25,
CH-1800 Vevey, Switzerland Tel. +41 21 922 13 00 Fax +41 21 921 23 53 www.martinig.ch
Editor: Franco Martinig; e-mail franco@martinig.ch ISSN 1023-4918
Free subscription: www.martinig.ch/mt/index.html