Risk-Based E-Business Testing

Part 1
Risks and Test Strategy

There are five main risk areas in E-Business (EB) system development. These risks relate to usability, performance, security, availability and functionality. These risks are not new. What is new is that functionality may not be the biggest concern and the shift in risk from functionality towards non-functional issues is dramatic. Most EB systems are simple (at least from the users' perspective) and the four non-functional areas present a challenge to business and testers.

The pace of development on the web is incredibly quick. 'Web-time' describes the hustle that is required to create and maintain EB momentum. Unfamiliar risk areas, rapid development, uncontrollable user base, new tools and techniques present the tester with a real headache.

What are the imperatives for EB testers? To adopt a rapid response attitude. To work closely with marketers, designers, programmers and of course real users to understand both user needs and the technical risks to be addressed in testing. To have a flexible test process having perhaps 20 different test types that cover each of the most likely problems. To automate as much testing as possible. These are the main planks of our EB Test Strategy.

Whether homegrown or proprietary, the essential tools are test data and transaction design; test execution using the programmer and user interfaces; incident management and control to ensure the right problems get fixed in the right order. Additional tools to validate HTML and links, measure download time and generate loads are all necessary. To keep pace with development, wholly manual testing is no longer an option. The range of tools required is large but thankfully, most are now widely available.

This paper presents an overview of the risks of E-Business and a framework for creating an E-business Test Strategy. The second part of this two-part paper will describe a collection of E-Business techniques and tools in detail.

1 INTRODUCTION

1.1 E-Business, E-Commerce and Everything Else

Before we go any further, let's get some definitions clear. The E-phenomenon is immature and vendors are generating new E-definitions thick and fast. We've taken, as at least an independent authority, the whatis.com definitions:

E-Commerce The buying and selling of goods and services on the Internet, especially the web.

E-Business The conduct of business on the Internet which includes:

- buying and selling plus servicing customers (internet)
- collaborating with partners (extranet)
- internal work and information flow (intranet).

1.2 Current State of E-Business Testing Experience

At a recent conference (February 2000) I asked a group of 190 testers a series of questions and asked for a show of hands. The table below summarises the responses:

Question	*% Saying Yes*
Use the web regularly?	100
Bought products or services online?	50
Experience of working on an E-Commerce or E-Business system that is now in use?	<5
Planning to build an E-Commerce system?	25
Planning to build an E-Business system?	30
Experienced problems when using a web site?	100

What interpretation can we put on these figures? Perhaps that the 'first generation' of E-Business systems did not involve the professional testers employed by user IT organisations. Perhaps these web sites were not tested adequately at all?

People abandon Web sites for all of the following reasons:

- Sites are difficult to use.
- Sites are too slow to respond.
- Sites were not trusted to be secure.
- Web pages broke (page not found etc.)
- Goods or services ordered were late, incorrect or never arrived.

Reference 1, Winning the On-Line Customer, Boston Consulting Group, (www.bcg.com) presents some startling statistics concerning the quality of E-Business systems, the response of users and the potential impact on suppliers. For example:

- 4 out of 5 E-Commerce purchasers have experienced failures.
- 28 percent of all online purchases fail.

These statistics make sobering reading. Not only could a poor E-Business site fail to gain more business. They could affect the business done by your bricks and mortar stores.

1.3 'Web time'

It might be regarded as hype, but there is no denying it, in some environments, 'Web time passes 5-7 times as fast'. What does this mean?

The notion of Web time could be summarised as follows:

- There are significant technology shifts every six months. The products you are using today may not have existed six months ago. The developers using these products are inexperienced.
- In some organisations, the E-Business projects may be owned and managed by non-computer experts. Software development is out; web publishing is in.
- Those working in the dotcom environments experience this phenomenon particularly acutely. The sprint to market dominates every other objective. There are many precedents that suggest that in an immature market, getting to market first means that you may dominate the market. In this kind of environment, does quality come second?
- There are obvious cultural, practical, technical and schedule obstacles to implementing thorough quality assurance and testing practices.

1.4 The E-Business Testing Challenge

Testers are used to working in time-pressured projects, without requirements, using unfamiliar technology in cultures that prefer to code first and plan second. But testers have rarely encountered all of these difficulties simultaneously.

This is the E-Business Testing challenge:

- Identify and address risks quickly and gain consensus on the overall test approach.
- Develop and implement flexible test strategies that address the risks of most concern.
- Devise new test techniques and methods that address the particular constraints of E-Business technologies.
- Make best use of test automation in every area of test activity.
- Provide thorough test evidence to help stakeholders to make the correct release decision.

1.5 Risk-Based Approach to Testing

If we believe the computer press, the E-Business revolution is here; the whole world is getting connected; that many of the small start-ups of today will become the market leaders of tomorrow; that the whole world will benefit from E-anyWordULike. The web offers a fabulous opportunity for entrepreneurs and venture capitalists to stake a claim in the new territory – E-Business. Images of the Wild West, wagons rolling, gold digging and ferocious competition over territory give the right impression of a gold rush.

Pressure to deliver quickly, using new technology, inexperienced staff, into an untested marketplace and facing uncertain risks is overwhelming. Where does all this leave the tester? In fast-moving environments, if the tester carps about lack of requirements, software stability or integration plans they will probably be trampled to death by the stampeding project team.

In high integrity environments (where the Internet has made little impact, thankfully), testers have earned the grudging respect of their peers because the risk of failure is unacceptable and testing helps to reduce or eliminate risk. In most commercial IT environments however, testers are still second-class citizens on the team. Is this perhaps because testers, too often, become anti-risk zealots? Could it be that testers don't acclimatise to risky projects because we all preach 'best practices'?

In all software projects, risks are taken. In one way, testing in high-integrity environments is easy. Every textbook process, method and technique must be used to achieve an explicit aim: to minimise risk. It's a no-brainer. In fast-moving E-Business projects, risk taking is inevitable. Balancing testing against risk is essential because we never have the time to test everything. It's tough to get it 'right'. If we don't talk to the risk-takers in their language we'll never get the testing budget approved.

So, testers must become expert in risk. They must identify failure modes and translate these into consequences to the sponsors of the project. "If xxx fails (and it is likely, if we don't

test), then the consequence to you, as sponsor is...” In this way, testers, management, sponsors can reconcile the risks being taken to the testing time and effort.

How does this help the tester?

- The decision to do more or less testing is arrived at by consensus (no longer will the tester lie awake at night thinking: “am I doing enough testing?”).
- The people taking the risk make the decision consciously.
- It makes explicit the tests that will not be done – the case for doing more testing was self-evident, but was consciously overruled by management.
- It makes the risks being taken by the project visible to all.

Using risk to prioritise tests means that testers can concentrate on designing effective tests to find faults and not worry about doing ‘too little’ testing.

What happens at the end of the test phase, when time has run out and there are outstanding incidents? If every test case and incident can be traced back to a risk, the tester can say, “At this moment, here are the risks of release”. The decision to release needn’t be an uninformed guess. It can be based on an objective assessment of the residual risk.

Adopting a risk-based approach also changes the definition of ‘good’ testing. Our testing is good if it provides evidence of the benefits delivered and of the current risk of release, at an acceptable cost, in an acceptable timeframe. Our testing is good if, at any time during the test phase, we know the status of benefits, and the risk of release. No longer need we wait a year after release before we know whether our testing is perfect (or not). Who cares, one year later anyway?

1.6 The Shift in Project Risks

In a recent IT project, Figure 1 Risks in a recent IT project, we conducted an early risk assessment to help the testers prioritise the testing and we identified 28 risks of concern. Of these, 21 related to the functionality of the product. This is typical of a traditional system development.

In a recent E-Business project, we identified 63 product risks of concern. In Figure 2 Risks in a recent E-Business project, only 10 of the risks related to functionality. In all E-Business projects, the issues of Non-Functional problems such as usability, browser configuration, performance, reliability and security dominate people's concerns. We used to think of software product risks in one dimension (functionality) and concentrate on that. The number and variety of the risks of E-Business projects forces us to take a new approach.

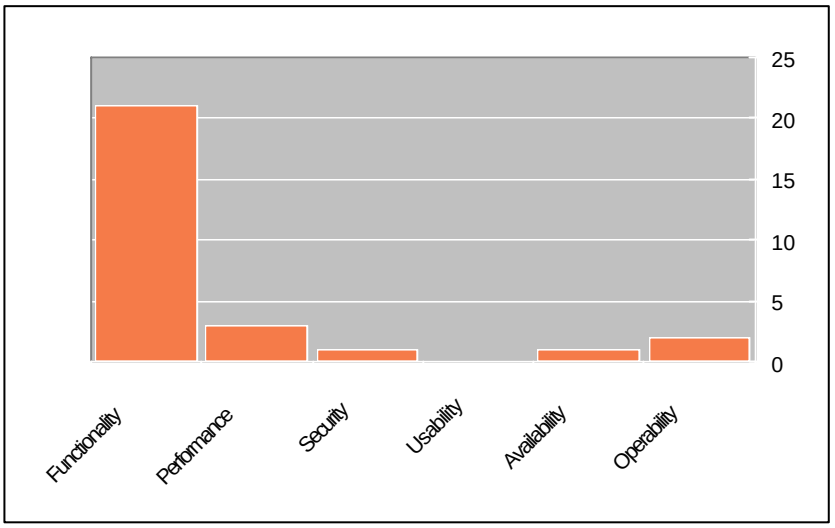

Figure 1 Risks in a recent IT project

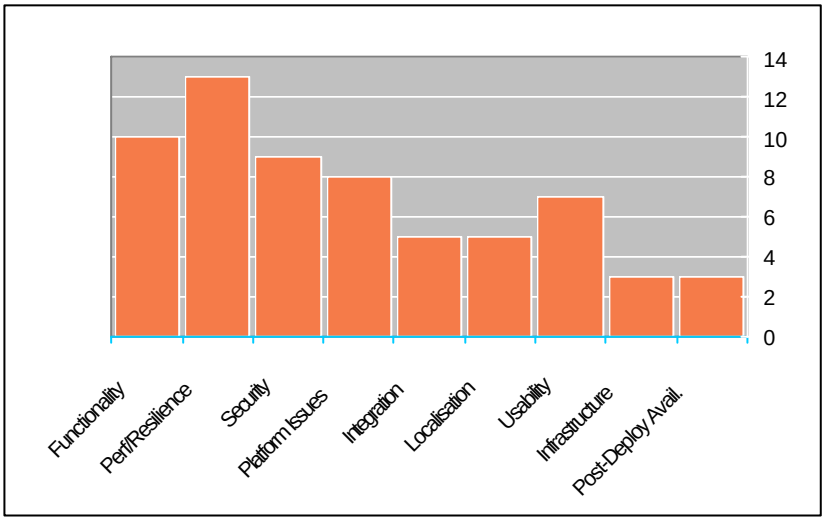

Figure 2 Risks in a recent E-Business project

There has been a shift in risks associated with E-Business projects and this forces us to reconsider how we address risk in testing. We need to spend time identifying the risks to:

- Help us prioritise what we need to do in our testing and
- Raise the visibility of the risks that are being taken by the project.

2 WEB RISKS

2.1 Web Failures

To help us understand the nature of risks facing the E-Business project, here are a few examples of problems reported recently on both sides of the Atlantic.

- A book publisher found that its system sent out books with blank labels because the address information was not carried through to the dispatch department, although customer credit cards were being debited.
- Many web sites advertise prices that include postage and packing. There are several reports of sites that assumed customers would be based in the home country. However, anyone in the world can access these sites and place an order to be delivered half way across the planet.
- A recruitment consultancy invited applicants to post their résumés on the web site but didn't provide facilities to manage them. They received thousands, but ended up trashing them.
- An airline advertised cheap tickets, but didn't remove the web pages when the promotion ended. Customers continued to demand cheap tickets. One successfully sued the airline when they were refused the special deal.
- A health-food wholesaler offered trade prices on their web site, but failed to direct retail customers to their local outlet. How many folk want 100 boxes of dried apricots at once?

2.2 Web Site as a Retail Store

It is a convenient analogy to compare a web site to a retail store. Just like a high street store, anyone can come in to browse or buy. The store is open to all including minors, foreigners, hackers, crooks and terrorists.

If your site gives good service, customers will come back again and again. If the buying experience is good, they will remember and use your store regularly. However, they won't come back if:

- The door is locked (your servers are down).
- It is difficult to find the product required (the product catalogue is out of date, incomplete or inaccurate).
- They get easily lost in the store (the site is difficult to use).
- Service is slow; there is a queue at the checkout (performance is poor).
- They feel that they can't trust the staff (the site feels insecure).

One of the most disturbing findings of the BCG report (reference 1) was that many customers wouldn't give you a second chance. Where a user experienced problems at a web site:

- 28% said they stopped shopping at that web site.
- 23% said they stopped buying at that web site.
- 6% said they stopped buying at the company's bricks and mortar stores.

The message to E-Business companies is clear: Improve the reliability and usability of your web site and give the customer a comfortable, fast, efficient and secure buying experience.

2.3 But Many of the Risks are Outside Your Control

Here is a summary of the risks that threaten your EB system. Many are outside your control, but may affect the customers' experience.

Unlimited potential users

There are virtually an infinite number of users who could visit your web site, browse and buy your products. You servers may be perfectly able to support 1000 visitors per hour, but what if your marketing campaign is exceptionally successful? Could your technical infrastructure support 5,000 users, 10,000 or 10 million?

Dependency on Internet Service Providers (ISPs)

Many web sites are hosted on servers operated and maintained by ISPs. Can you rely on these companies to provide the support you may have grown use to in your internal data centres?

You have no control over client platforms

Most users of the web have 'traditional' hardware such as PCs, Macintosh or Unix workstations. However, many users of the Internet in future will use technology that is either available now on the horizon. WebTV, mobile phones, and PDAs are all being promoted as alternatives to existing browser based devices. Cars, fridges, Internet kiosks and many other Internet capable devices will be marketed in the next few years. Could your current web developments accommodate these future platforms?

You have no control over client configuration

Just like the new Internet-ready devices that have different screen sizes and resolutions, your web site may have to accommodate unusual user configurations such as very large or very small screens. Further, PCs and browsers can be configured with foreign character sets. Will your back-end systems and databases accept these unusual local character sets?

You have no control over client software

Visitors to your web site may have any operating system, any version and in any condition. If you have children who share your home PC, you'll know all about the problems caused by curious kids installing games, deleting files and fiddling with control panels. But beyond the state of the operating system, the user has a large choice of browser technologies. Microsoft Internet Explorer and Netscape dominate the market, but there are many, many other products available. Visit http://browserwatch.internet.com for a comprehensive list of browsers. There are over 20 that run under Windows, some are non-graphic (e.g. Lynx). Other obscure browsers, such as Cello (http://www.law.cornell.edu/cello/), for example, run on machines with extremely limited configurations (2MB of RAM on a 386SX-16!)

Your users may not have required plug-ins

Does your application demand that users have plug-ins? Common or not, not all users will have the required plug-ins (or versions) required. If your site depends on Adobe Acrobat, Flash! or an up to date version of the Java Virtual Machine to operate correctly – not all users appreciate being asked to download the software required. Some may refuse. In this case, will your application still work? How will your user know this? What happens if they refuse the download, or it fails to install?

The context of web transactions is a risk area

The HTTP protocol is stateless. What this means is that under normal conditions, every request to a web server is treated as an independent, autonomous, anonymous transaction. The server cannot link series of messages from a single client machine without some further information. Typical methods are cookies (see below) and hidden fields on HTML forms. Cookies can be rejected or time out. Web transactions can be interrupted through loss of network connection, a visit to another web domain or using the browser back and forward buttons.

Cookies can be a problem

Cookies are a common method for holding web site-specific data on the client machine. Cookies are typically small amounts of data written to the client machine's hard drive to hold personalised data or information that contains data to be carried through web pages as they are navigated. However, the mainstream browsers can be configured to warn the user when these are being requested by the web sites visited. Some users may allow and ignore them, some want to be reminded when they are invoked, some will reject them. Will your application work, if cookies are rejected?

Network connections

If your web site provides business-to-business services, it is likely that your customers access the web via a high-speed local network and the customer's site may have a direct connection to the Internet. In this case, the download speed of web pages, even large ones may never be a problem. However, business people working from home, and the vast majority of consumer customers will use a dial up connection and modem. The relatively slow speed of modems dramatically slows the pace of page downloads, particularly those containing large images. The perceived 'slow speed of the web' is usually due to over-large images that have not been optimised. This is a very serious consideration – slow web sites are unpopular and many people lose patience and leave them prematurely.

Firewalls may get in the way

There are some situations where the configuration of a customer's firewall can cause your site to be unusable or inaccessible. Whether your site is 'banned' by the customer network administrator, or the settings of your web servers conflict with the remote firewalls, your site may be unusable.

Anyone can 'walk in'

Just like a real shop, your web site is open to the public. Users can be experts or hackers but they are always unknown to you. Anyone on the web is a potential customer, but you don't see them and they don't see you. Some may be under-age. How can you tell if a 10 year old places an order with their parent's credit card? How would you know that you were selling alcohol to minors?

Usability is now a prime concern

Your site is unlikely to be documented. Your users are almost certainly untrained. If the site is hard to use, they'll go elsewhere. What is your experience of web sites? If you find it difficult to navigate, confusing, unpleasant to look at, time consuming, slow on more than one occasion, you may have just given up. Maybe you didn't even reach the functionality that was designed to help you place an order or find information. The user experiences good or appalling usability at the front door of the web site so usability is now a make-or-break issue for your EB developments.

Internationalisation

Once your web site is up and running, anyone on the planet who has the technology can visit and use your site. If you are offering an international service or delivery this is a fabulous opportunity. However, people in foreign countries may have to use their own language. Many will not appreciate 'obscure' address formats. They may wish to pay in local currencies. They may have their own local tax arrangements.

3 EB TEST METHODOLOGY AND THE W-MODEL

The EB test methodology that we will present in this paper is a natural extension of our W-Model view of testing.

To many software engineers and development managers, testing is the last phase of the development life cycle. However, waiting until after executable software has been built is the most costly and least effective way of performing testing.

The belief that tests can only be run against executable software is, frankly, wrong. There are test techniques that can be applied throughout the entire development life cycle, even as early as the requirements gathering stage. These techniques are not only cheaper than dynamic testing, but they also avoid wasting money on building the wrong system. Defect prevention is cheaper than defect correction.

Testing is least costly and most effective if it is performed throughout the whole life cycle, in parallel with every stage of development. This strand of testing in parallel with development is represented in the W model.

For those who are familiar with the V model, the W model is a natural evolution. The V model illustrates the layered and phased nature of software testing, but lists only dynamic test stages like unit and system testing. Some variations also include early preparation of test cases. The W model, by contrast, supports testing of all deliverables at every stage of the development life cycle.

The W-model promotes the idea that for every activity that generates a project deliverable, each of those deliverables should have an associated test activity. This is certainly not a new idea. Where it differs from the V-model is that it promotes both static testing of early document or code deliverables and dynamic test stages of software deliverables.

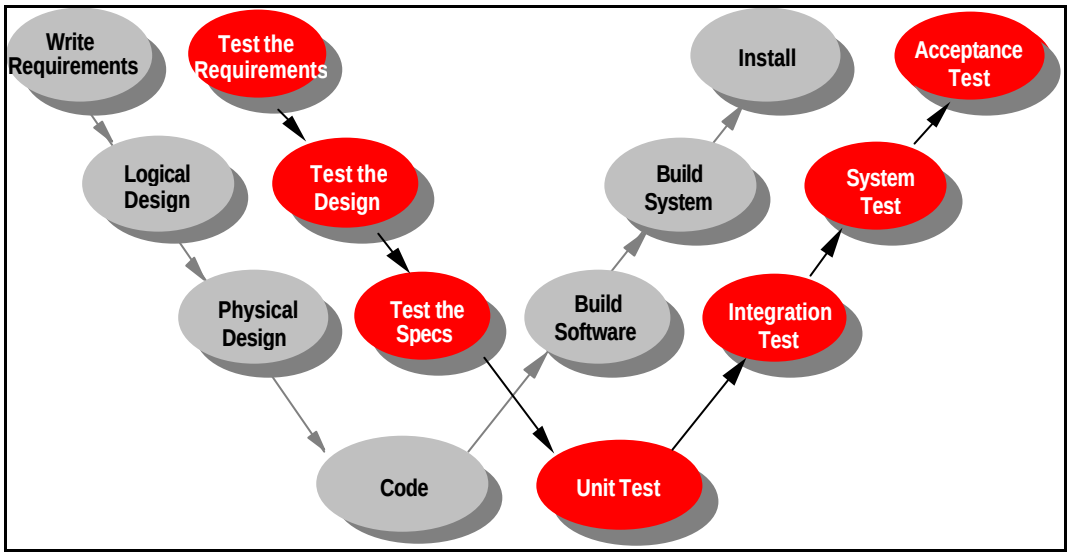

Figure 3 The W Model

Evolutif doesn't suggest that dynamic testing is obsolete, only that testing has to start right up front during requirements definition. The effect is a dramatic reduction in faults found in dynamic testing (the faults are removed earlier, before the wrong code is written), fewer faults found in live running, and certainty that the delivered system meets requirements.

Total Testing™ is Evolutif's implementation of the W model. It encompasses a range of static test techniques – like behaviour analysis, requirements animation, scenario walkthroughs, document reviews and inspections – as well as dynamic testing and early preparation of dynamic test cases.

Figure 4 Static tests in the lifecycle identifies the most commonly available static tests that are useful in the testing of requirements, designs, code and test plans, perhaps).

Figure 5 Dynamic tests in the lifecycle presents the most commonly used dynamic tests that are appropriate for testing components, sub-systems or entire systems. This includes of course, the many non-functional types of testing.

The value of the W-model approach is that it focuses attention on all project deliverables. By matching the deliverables with product risks, the types of testing required to be incorporated into the test strategy can be quickly organised into the test process.

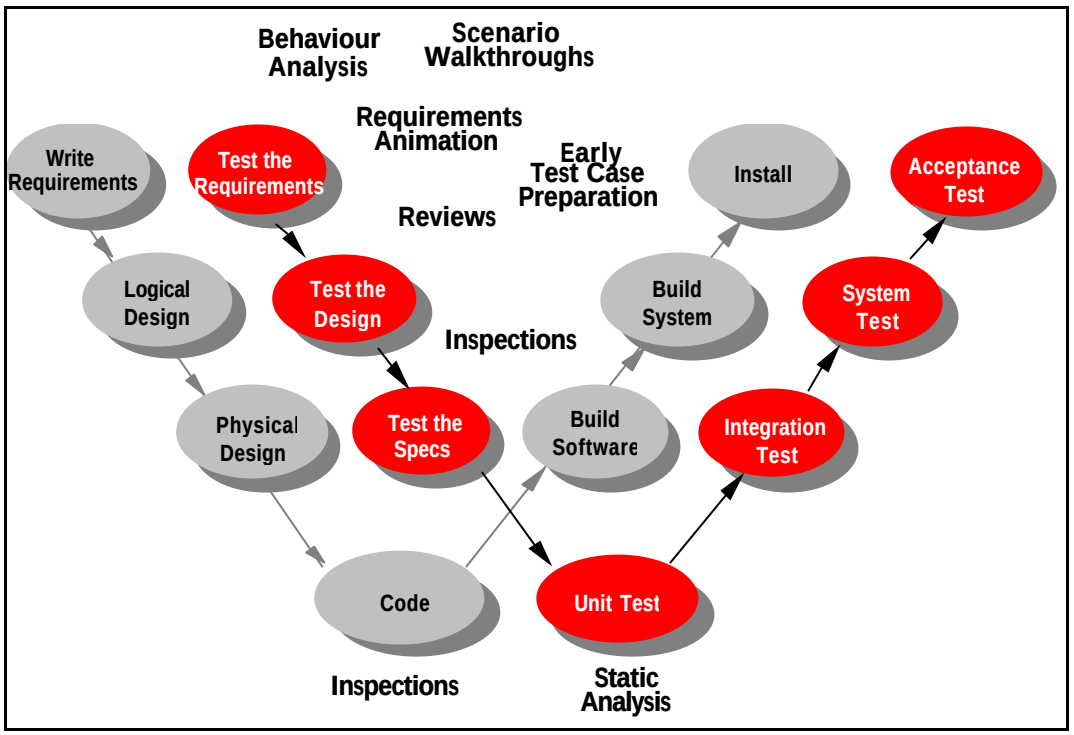

Figure 4 Static tests in the lifecycle

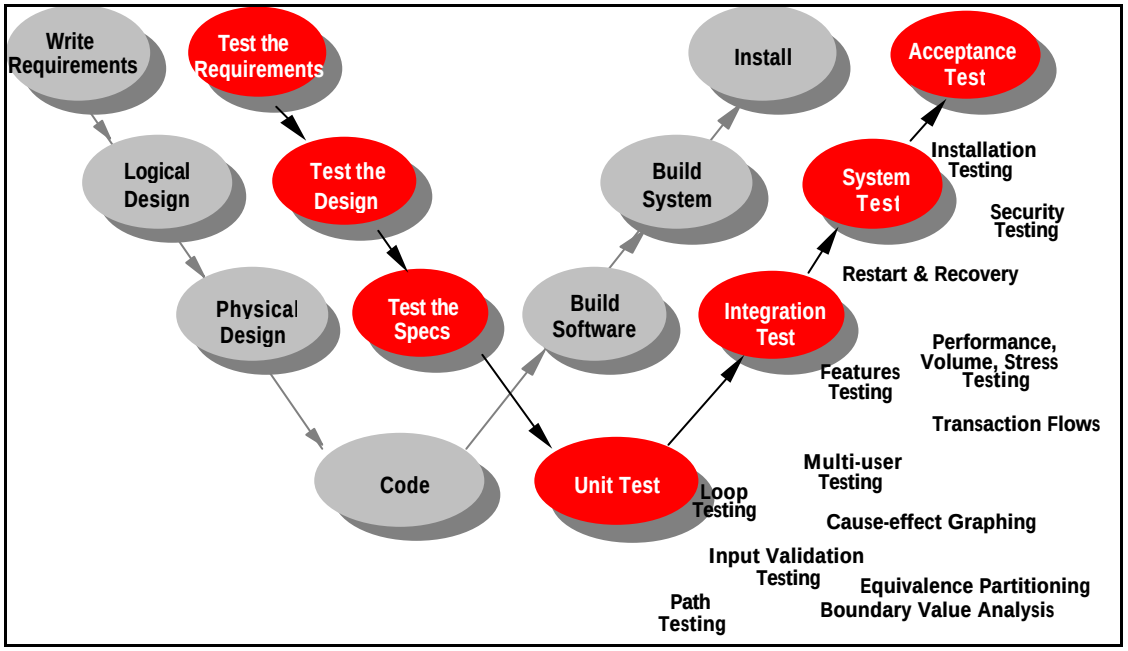

Figure 5 Dynamic tests in the lifecycle

We have used the W-model approach to identify types of test for the various deliverables of E-Business projects and to build a collection of techniques structured into a generic testing framework that we believe should help testers address the E-Business testing challenge.

4 E-BUSINESS TEST STRATEGY

4.1 Specialist Knowledge and the New Test Techniques

Testers need more than a minimal technical knowledge to plan, specify, execute and analyse tests of Internet based systems. The technical details of how the web works are easy to obtain – there are now hundreds of books covering HTML, CGI, Java, ASP and the rest of the many concepts, terms and technologies. The knowledge seems easy to acquire, but where does an E-Business tester's knowledge start? The more difficult question is, "where does it stop?"

The list of topics listed in Appendix A are a minimum set of terms and concepts that the E-Business tester should be familiar with. Beyond the most basic issues, there are a number of more advanced topics that the tester should have at least a passing familiarity.

Some of the test techniques need a perhaps programmer's level of knowledge. This is an unfortunate consequence of working in a rapidly changing technical environment.

In particular, the tester needs to know where sub-system (developer) testing stops and system testing starts.

4.2 Testing Considerations

Here are some broad issues that need to be considered when formulating the E-business test strategy. If testing is to keep pace with development, we need to squeeze the testing into the develop/release cycle.

Automation-focused approach

Design tests to be automated from the start. There won't be enough time to automate manual tests later. Plan all tests once only and reuse as regression tests. Where there is dramatic change in the user interface, consider using test drivers (below) to do the bulk of functional tests.

Consider working with developers to help them automate their tests and your own. If you are all using the same tool, the transition between development and system testing will be smoother. Developers can more easily reproduce the faults reported by later system testers if there is common test technology.

Developer testing

Consider tying in tests to developer code check-in/check-out procedures. Firstly, this will ensure that they do some testing and secondly, a trusted set of regression tests will be maintained throughout the development.

Reference 2, Extreme Programming Explained, Kent Beck, ISBN 0-201-61641-6, promotes a set of disciplines for developers that encourage them to do significantly more and better testing. Pair programming, continuous automated testing, permanently maintained regression testing all combine to improve the development test process if (and it's a big if) the developers are prepared to work this way.

Consider asking the developers to use a variety of browsers (and versions) for their testing. Rather than using the company's chosen browser, encourage them to use Internet Explorer, Netscape or their favourite browser because using a variety of browsers can help to flush out compatibility problems across the different browser types. You may be able to eliminate later configuration testing!

Consider using test drivers

From the point of view of the user interface, Web applications are often relatively simple compared to traditional client/server or mainframe based systems. However, the user interface of Web applications is sometimes the very last thing to be developed and the

integration of automated tools with browser-based applications is not always perfect, so consider using test drivers to execute functional tests of server based code.

The general recommendation with Web applications is to place as much functionality on servers, rather than the client browser. Where the Web is being used as a front-end to existing legacy systems, the testing of the back-end systems is as complicated as ever. Consider testing the back end functionality using the application programmer's interface (API). Test drivers are often very simple programs that accept test data, construct the call to the server-based code, execute the transactions and store the results for later evaluation. Some drivers can be enhanced to perform simple comparisons with previously run benchmarks.

There are a limited number of proprietary tools that can help, but you should consider writing your own in a few hundred lines of Visual Basic, Perl or C++ etc.

4.3 Using a Test Process Framework to Build Your Test Strategy

The Testing Framework outlined in Table 2 E-Business Test Process Framework, is an attempt to provide a framework for testers to construct their own test process from an assessment of risks faced by their project. (The risk assessment methodology that we use in our projects is not described in this paper). Based on an assessment of E-business product risks, the test types listed in the first column would be selected to address each risk in turn. The structure of the test process table is intended to help you to construct the detailed test stages from the test types.

Each risk can be transformed into a test objective to be addressed by the test types.

Test types are grouped into five categories

To address the risks of the E-Business project, we have identified 20 distinct test types. Each type of test addresses a different risk area. The purpose of the framework is to help you to construct your own project-specific test process. The test types are grouped into five main categories:

- Static testing
- Test Browsing
- Functional Testing
- Non-Functional Testing
- Large Scale Integration.

These reflect the nature of the test activities themselves. For example, static tests are those relating to inspection, review or automated static analysis of development deliverables.

Tests can be static or dynamic

Test types can be static or dynamic. There are actually only four static test types.

Essential Testing Priorities

Interesting aspects of E-Business projects are the extremes of project pace. In mature companies trying to establish a web presence, the existing IT department may be responsible for the development and they may follow their standard, staged development and test process.

However, some projects operate in Web Time. If you are working in a dotcom environment, where speed to market is the first priority, you may have to work under such pressure that the testing has to be squeezed into a few days or even a few hours. Release cycles may be daily. How can the traditional test process be fitted into this kind of schedule?

It is clear that the tester must be prepared to adopt test practices that complement these two very different scenarios. Our proposal is that a range of 'test priorities' be used to align with the project development process. Table 1 Essential Testing Priorities presents one possible way of looking at the most important attributes of your web site. They are introduced here to give you an impression of the stark choices you may face in your projects. We suggest that you consider the following four levels of test priority.

Testing Priority	*When to Use*
1. Smoke testing	▪ Can the application survive one user session without crashing? ▪ If you are under severe time pressure and have hours to do the testing, stick to smoke testing.
2. Usability	▪ If your site is hard to use, the user won't hesitate to leave your site before reaching the functionality. ▪ There are several techniques involving inspection, review and heuristic evaluation.
3. Performance	▪ If your site is usable, but slow, users may give up on the functionality and leave. ▪ Performance tests measure the speed of the site; flush out weak points, measures limits.
4. Functionality	▪ Functionality isn't always the biggest issue. On basic E-Commerce' sites, functionality may be simple and relatively easy to 'get right'. ▪ Functionality is obviously tested in all projects, but often the functionality can be proven during development and user evaluation.

Table 1 Essential Testing Priorities

The test types have been allocated a slot against the four test priorities mentioned earlier. Test types that fall into the Smoke testing column, are typically automated and should be the minimum set of tests considered to address the maximum number of risks in the shortest time. The tests that fall into the Functionality column are typically the tests that are most expensive and time consuming to implement and execute. The purpose of these columns is to help you to select the test types should be included in your test process.

Five stages of testing

We have found that the test types do not always fit into the traditional unit, link, system and acceptance test stages. So we provide a test structure that has five test stages that group the test types by the clear technical groupings:

- Desktop development testing (broadly, what the browser executes)
- Infrastructure testing (what runs on the servers)
- System testing (of the complete system in isolation)
- Large Scale Integration (with other systems)
- Post-deployment monitoring (retaining automated tests to monitor live sites).

Tests can be automated or manual

The A and M entries in the table denote whether the test can be automated:

- M – the test can only be done manually.
- A – the test can only be done using a tool.
- A/M – the test can be done manually or using a tool.

Tools are covered later in this paper.

Table 2 E-Business Test Process Framework

	Test Priorities					*Test Types Mapped to Usual Test Stages*				
Test Type	***Smoke***	***Usability***	***Performance***	***Functionality***	***Static/ Dynamic***	***Desktop Development***	***Infrastructure Testing***	***System Testing***	***Integration Testing***	***Post-Deployment Monitoring***
Static Testing										
Content Checking		Y			S	A/M				
HTML testing	Y				S	A/M				
Browser syntax compatibility	Y				S	A				
Visual browser validation		Y			D	M		M		M
Test Browsing										
Link checking	Y				D			A		A
Object load and timing		Y	Y		D			A		A
Transaction verification	Y				S	A/M		A/M		
Functional Testing										
Browser page testing	Y				D	A/M				
CGI component testing	Y				D		A/M			
Transaction Testing				Y	D			A/M		
Application System Testing				Y	D			A/M		
Internationalisation		Y			D	A/M		A/M		
Non-Functional Testing										
Configuration testing	Y				D	M		A/M	M	
Performance			Y		D		A	A		A
Soak Testing/reliability	Y				D	A	A	A	A	
Availability					D					A
Usability		Y			S/D			M		
Security				Y	D		A/M	A/M	A/M	A
Large Scale Integration										
External links/legacy system integration				Y	D		A/M		A/M	
End to end functionality	Y				D				A/M	A

4.4 Guidelines for Using the Test Process Table

Not all test types are appropriate or possible in all circumstances. We are not suggesting that you must commit to doing all the test types identified in the framework. Rather that you gain a consensus view on the risks of most concern, match this to your development process and use the framework to help construct your test process.

The sequence of tasks required to construct the test process is as follows:

1. Identify the risks of concern with the project stakeholders, management, users and technologists and prioritise those that must be addressed by the testing.
2. For each risk to be addressed, identify one, or more, test types that can help to address the risk. Be sure to document those risks that will **not** be addressed by the testing.
3. Using the text of the risk description, write up test objectives (e.g. "to demonstrate that the Web application operates reliably in a range of configurations") to show how this risk will be addressed.
4. Assign the test type to a test stage and nominate the responsible party within the project.
5. Estimate the time required to plan and execute the test.
6. Review the scope of the testing (what is in scope and what is out of scope), test objectives, stage definitions, responsibilities and estimates and refine.

The framework is a guideline to be considered and adapted - tailor it to your environment.

4.5 Post-Deployment Monitoring

Post-deployment monitoring is done after implementation in a production environment. The reason for including this in your test strategy is that production E-Commerce sites can fail, but it may be that none of your prospective customers will tell you. They'll simply go elsewhere.

Further, degradation of site performance may not be noticeable day by day, but using a tool, statistics can give early warning of performance problems while there may still be time to recover the situation. Hence, we recommend that some selected automated tests be re-used to monitor the site. Services aimed at monitoring your site remotely are now becoming available, so this is an alternative worth considering.

4.6 Other Test Scoping Issues

Which platforms will your software support?

What browsers and versions will your Web site support? Do you need to worry about no-frames browsers? Do you need to consider non-graphic browsers? These questions are important because they help you to determine the scale of regression testing across a range of browsers and versions. Even if you will support only Explorer and Netscape, which versions of these browsers should you test to ensure there are no configuration problems?

Propose a scope for the testing to be performed (and estimate the potential cost) and review that with the stakeholders, management and technical experts in the project. This will help to force a decision on what will actually be supported, and that may be much less than was envisaged by management. If you are going to test many of the browsers and versions, you will be committed to doing an awful lot of regression testing.

Consider and propose an alternative: that the developers work to the HTML standards supported by the chosen browsers. Tools can then be used to validate the HTML in your web pages. There number of browser platforms available is steadily growing. In the future, it looks increasingly likely that standards-based development and automated

HTML validation will be a more practical approach than expensive regression testing across a large range of platforms.

Which Web conventions should be adhered to?

Although adherence to conventions is never mandatory, following Web conventions can make a big difference to your users. If your web site behaves differently to 90% of other sites, it might look good superficially, but may cause your users so much hassle they stop using it.

For example, will your application support users turning off graphics? Dial-up users regularly switch off graphics to speed things up. Will the site work if users reject cookies or the cookies time out prematurely? Must users always have the required plug-ins? As a tester, you need to establish what conventions apply to you can test against these, and ignore other contraventions. The best thing about web conventions is that they are widely documented and the browser defaults support them. Testing becomes straightforward.

The Web Accessibility Initiative (WAI), The World Wide Web Consortium, (www.w3.org) guidelines paper is an ideal document for testers to work against. Any contravention can be reported as a fault to the developers. In that way, the document is a good baseline (invaluable where other requirements may be lacking). So, get your project either to obey some or all conventions, and use a baseline to test against, or eliminate web conventions from the considerations in your test plan.

5 REFERENCES

1. Winning the On-Line Customer, Boston Consulting Group, (www.bcg.com)
2. Extreme Programming Explained, Kent Beck, ISBN 0-201-61641-6
3. The Web Accessibility Initiative (WAI), The World Wide Web Consortium, (www.w3.org)

This paper is Part 1 of a two-part document. In Part 2, the details of the various test types introduced in this paper are described in detail. Automated tools that support the various test types are also covered.

APPENDIX A - ESSENTIAL INTERNET CONCEPTS FOR THE TESTER

HTTP	Hypertext Transfer Protocol - the way that client browsers and web servers communicate
URL	Unified Resource Locator e.g. http://www.evolutif.co.uk/subdir/index.html
HTML	Hypertext Markup Language Web pages comprise HTML tags and ASCII text
CGI	Common Gateway Interface – the mechanism by which HTML pages can execute programs that reside on web servers. CGI are Perl, C++ or other programs that execute on the server. Using HTML forms, users can send data and have it processed on the server
Cookies	Small quantities of data stored on the client's hard drive at the request of a web site. The web site can 'read' the cookie on a later visit. Used to carry data through Web transactions.
How a web page works	▪ You select a target URL in your browser by using the URL window or clicking on a link. ▪ The browser sends a HTTP request for the HTML file to the site defined by the URL. ▪ The server processes the request; sets up a connection, sends the HTML file and closes the connection. ▪ Browser interprets the HTML file and displays it according to HTML standard.
JavaScript/VBScript	Programming languages (cut down versions of Java and Visual Basic) used to provide functionality within a web page on the client or on servers.
ASP	Active Server Pages – a Microsoft technology that allows VBScript or JavaScript embedded in HTML pages to be executed on servers.
Applets	Programs normally written in Java that are downloaded from servers and executed within the browser.
Servlets	Programs normally written in Java that are executed on servers.
The web is stateless	After processing an HTTP request, the server forgets everything about the transaction. Developers need to maintain context through business transactions so use cookies, hidden form fields to do this.