

Guidelines

number 265

FOR IT MANAGEMENT

Web Application Development

CONTENTS

page

1. INTRODUCTION	3
2. CAPTURING THE INITIAL REQUIREMENTS	3
2.1 Choosing the application and site type	3
2.2 Use case definitions	3
2.3 Functional specifications.	4
2.4 Usability requirements	4
3. HIGH LEVEL DESIGN: WEB APPLICATION ARCHITECTURES	4
3.1 System architecture.	5
3.2 Application architecture	5
3.3 Infrastructure architecture	5
3.4 Information architecture	6
3.5 Security models and security architecture	6
3.6 Developing future releases.	7
4. DESIGN METHODS AND DEVELOPMENT METHODOLOGIES	7
4.1 Development methodologies	7
4.2 Site flow.	7
4.3 Object design	7
5. DEVELOPMENT TOOLS AND ENVIRONMENTS	8
5.1 Open source technologies (Linux, Apache, MySQL, Perl/PHP)	8
5.2 Microsoft technologies	8
5.3 Java technologies	9
5.4 Other technologies	10
6. BUILDING AND DEPLOYING WEB APPLICATIONS	10
6.1 Prototype often	10
6.2 Usability studies.	11
6.3 Development test frameworks.	11
6.4 Staging server	11
6.5 Deployment and systems integration	11
6.6 User acceptance testing.	11
6.7 Soft launch.	12
6.8 On going monitoring	12
7. ONGOING ACTIVITY	12

© Current members of NCC from time to time are permitted to copy this Guideline up to a maximum of four times for internal purposes only. Full acknowledgement must be given to NCC at all times. All other copying is prohibited unless written permission is granted.

1. INTRODUCTION

While the dot com bubble has well and truly burst, web technologies haven't gone away. In fact, more and more organisations are using them to add business value – both internally and externally.

Generally a web application is seen as an externally facing transactional service that is accessed through a web browser. However, in practice this means anything from a simple online catalogue service to an online banking service that duplicates the functionality of a standard branch. The content management architectures used to deliver a modern informational site like BP.com make it possible to describe it as another form of web application.

Inside the business the corporate intranet is slowly changing into the enterprise portal. Here a web application is used to deliver personalised information to users, as well as acting as a gateway into a web-front-ended line of business applications. Acting as a single point of call, the enterprise portal is an important driver for the development of web applications within the framework of a modern business.

While some will describe web application development as a whole new paradigm, liberally scattering buzzwords throughout a presentation, in practice web application development is very similar to traditional application development. The key difference lies in the architectural approaches used. It's very difficult to put a web-front-end on a traditional monolithic application – or even a client-server solution. Instead of the standard single user interface, a web application needs to work through a set of pages hosted in a browser.

This approach tends to lead to applications that are developed as a set of communicating components. Using web pages as the sources of application events, and as the place where results are displayed, web applications collect together these components into applications that closely resemble the three tier model: separating presentation from business logic and data.

2. CAPTURING THE INITIAL REQUIREMENTS

Before any development work begins, it is important to get the foundations in place. Like any large scale development programme, a web application needs to start from a clear set of specifications.

2.1 Choosing the application and site type

The first question that needs to be asked is: what type of application is being built? Is it an online store, or is it a B2B information sharing service? Is it an internal knowledge management solution or is it a customer relationship management solution? Then it is important to assess the audience, and their access methods. Only once a clear statement of the application's purpose has been articulated, does it become possible to start developing the initial requirements.

While it may seem easy to define the type of site, this can actually become a complex process. Everyone involved with the project will have a different idea of what is intended, and it is only once all these views have been captured and analysed that a clear picture of the site aims can be defined.

This process can be internal, or can involve members of the target audience. Certain types of site – especially those that require interactions with users from outside the organisation – will need to begin with an understanding of the user needs before any definition process can begin.

This process will end with a definition of the application scope, a listing of desired features that can be prioritised by the various development teams. An initial scope document will aid in planning not just the first release of an application, but several subsequent releases.

2.2 Use case definitions

Any web application will be designed to meet a specific set of business needs. As the development process begins, one of the first tasks will be to capture how the application is intended to be used. Business analysis techniques can be used to capture the usage requirements from both inside and outside the organisation. Once captured, the information can then be mapped into a series of use cases, which can be used to provide the basis for detailed functional specifications.

Written in standard English, with clear diagrams, use cases are a powerful tool for developing web applications, as they can be used to:

- Document the business process.
- Identify possible collaborative business areas.
- Separate business processes into functional system areas.
- Serve as requirements documentation for system development.
- Identify possibilities of system or component reuse.
- Categorise requirements (e.g. state of implementation, functional system, etc.).
- Rank requirements (e.g. level of importance, risk, level of interoperability, etc.).

2.3 Functional specifications

Once the use cases have been defined the functional specifications can be written. These should document the use cases in detail, along with any interface requirements. This is a critical document, as it will form the basis of any development process.

The functional specifications will need to be agreed by the design team and any project governance body, since they will act as the basis for any user acceptance tests. If a third party development organisation is used, the specification document will need to be part of any agreed contract, and variations from it will need to be managed by an effective change control procedure.

In some cases functional specification documents may also include site flow and indicative page layouts. If a web application is intended to integrate with a third party, the specifications will also need to define acceptable message formats. As third party integration is one of the most complex parts of any application development, especially if business processes are involved, it is important to define roles and responsibilities (as well as specifications) early in the development cycle.

2.4 Usability requirements

Usability needs to be considered at an early stage. As the functional specifications may include site flow and page wire frames, these will need to be developed with the end user in mind. Any usability requirements will need to be included in the functional requirements document.

The process of developing usability is more than just design. It will involve user research, as well as development of design frameworks and principles. The user research process will include interviews and user observation. One side effect of the user research process is the identification of business needs that may have escaped traditional analysis techniques. However, this process can be time consuming – especially if a full scale user segmentation study is required.

3. HIGH LEVEL DESIGN: WEB APPLICATION ARCHITECTURES

The architectural approaches taken when designing web applications will vary depending on the required functionality. However there are certain key architectural approaches that are common to all developments:

- System architecture.
- Application architecture.
- Infrastructure architecture.
- Information architecture.
- Security architecture.

3.1 System architecture

The system architecture is the starting point for the development process. Often referred to as a 'high level architecture', it allows the development team to define how they will meet the functional requirements. Once complete, the systems architecture can be used to develop the applications architecture and infrastructure architecture of the site.

Initially drawn as block diagrams, a system architecture should indicate how the various components of a web application will be implemented, indicating rough numbers of servers, possible data requirements, interfaces and possible candidate applications. Data flow requirements can also be included in the system architecture.

Once defined, the system architecture can be used to inform the teams developing the application and infrastructure architectures. In some cases systems architectures will be more detailed, and will be used to document overall design patterns used by the application architectural team.

3.2 Application architecture

The heart of the high level design process is the application architecture. It is here that the architectural team not only defines the platforms used to support the web application, but also begins to break down the functional requirements and starts the process of designing the application components.

The application architecture is intended to define the road map for the whole of the application development. It will compare candidate platforms and packages, and deliver a recommendation of the tools and technologies that will be used to deliver the application. This will include application servers, databases and web servers.

Functional requirements and use cases will be used to define the key components to be utilised within the application, along with interfaces and the overall data model. This will allow an assessment of the application complexity, and will feed into the platform definition process, as well as providing a basis for further detailed design.

3.3 Infrastructure architecture

The infrastructure architecture of any web application is made up of the hardware and network design used to support any application. As a result, it requires significant work in order to ensure that performance criteria are met, and that there is enough reliability and scalability built into the design to support proposed service levels and planned growth.

It is important to develop an initial network design as soon as the functional requirements have been defined. This will allow the development team to have a target environment that can be used to aid in application design decisions. The network design should include, where possible, firewalls and load balancing appliances, as well as a basic list of hardware requirements for servers and storage.

Any decisions on server requirements will need to be made in conjunction with the team defining the application architecture. Server requirements will need to support any products used to develop the application, along with bespoke code and data needs. Scalability and reliability will need to be considered here as well, determining failover and clustering procedures and architectures.

A useful tool for the infrastructure architecture team is a set of scaling spreadsheets. These allow architects to link system requirements with calculated bandwidth requirements, known application performance (from public benchmarks and from experience with similar tools), and the number of available servers. As changes are made to the application requirements, the spreadsheet can be re-run to investigate whether the infrastructure is still appropriate. This tool can also be used in conjunction with actual and predicted usage figures to determine when new capacity is required.

The infrastructure architecture will be closely related to the security architecture, and it is likely that a member of the infrastructure development team will be responsible for developing the overall security model. This will ensure that the resulting application is designed for security from the ground up.

3.4 Information architecture

The information architecture of any web application is critically important to the overall success of a site. As well as detailing the structure of a site, and the basic layout of pages, the information architecture can be used to define the overall site navigation model, as well as helping to develop a content strategy for any content-rich web application.

While often seen as a creative process, developing an effective information architecture is critical to the success of any web application. Without it sites can be launched that do not offer users a compelling experience, and as a result, don't receive the return visits that lie at the heart of the business model used to justify the investment in the web application.

One of the most important components of the information architecture is an understanding of how the web application is likely to be used – and the resulting navigation metaphor and layout guidelines. These will be used by both the development and creative team during the application development process.

3.5 Security models and security architecture

As web applications are often critical components of any business IT infrastructure – and often public facing – security is a major concern. The recent outbreaks of worms like NIMDA show that unsecured web servers and web applications are a risk to the rest of a business.

A key component of any web application design is its security model. This covers everything from application access control lists to the specifications of any firewalls used, along with policies and procedures. It's easy to fall into the temptation of keeping costs down by skimping on security, but that is most definitely a false economy. One key recurring cost will be system administration time, in monitoring the appropriate security announcement mailing lists, and in preventative maintenance.

The organisational component of any security model is critically important. If an appropriate security policy is not put in place, procedural errors can put a site at risk. Policies will need to detail specific roles and responsibilities for all users – including system and application administrators. Appropriate access to servers and management tools will need to be defined, along with rules that define how application components communicate.

Closely related to the security model is the security architecture. This defines the actual hardware and software used to implement the security policy. Technologies involved in the implementation of security architectures include firewalls, authentication services, entitlement management and network design. While most of these may be familiar, entitlement management tools are a relatively new entrant in the field, and should best be thought of as the point where access control and personalisation meet. With these services it is possible to give different users a different view of a site depending on what information they are entitled to view. This would allow separation of registered users from casual browsers, customers from partners, and users from administrators – without needing to provide different versions of a site, or a page, for each class of user.

Firewalls need to be implemented at any point in an architecture where trust levels change. Typically, in an externally facing web application these are best characterised as:

- Externally facing services (web servers, ftp, etc.).
- Web application services (application servers, etc.).
- Data services (databases, legacy systems, etc.).
- Management services (system administration tools, monitoring tools, log analysis tools, etc.).
- Third party gateway services (payment services, messaging services, etc.).

Trust will be low for externally facing services, as these can possibly be compromised, and high for internal data services, which may be critical line of business applications. Management service

layers and third party services are more complex, as they may interact with a web application at any point in its hierarchy of trust.

3.6 Developing future releases

How a web application evolves over time is critically important. While you may launch with release one, there's a whole raft of future releases to consider – the first of which may be less than six months away. Some popular sites, like Amazon.com add functionality regularly, with several major releases a year.

While the release scope for release one will have been determined before development begins, it is important to keep planning for the effects of additional scope in future releases. As a result, it is important to keep track of how functionality changes will affect the site. The site scaling tools used to determine the infrastructure architecture are an important tool at this stage, as they allow you to determine further server requirements, and to predict bandwidth needs as site scope changes.

This isn't the only tool you have. It's possible to use the assumed complexity of any requirement to determine the loading in application servers or on databases. Monitoring tools should be used to keep track of current usage, as well as the performance of application servers. Where custom tools have been used, developers should make sure that appropriate reporting and monitoring features are included.

When planning future releases, all the available information should be used to determine the requirements, and to define the release scope.

4. DESIGN METHODS AND DEVELOPMENT METHODOLOGIES

Once the high level design process is complete, the web application rapidly moves into detailed design and development. The detailed design will include application site maps, as well as object and data models. Appropriate development methodologies can be chosen at this stage.

4.1 Development methodologies

As web application development is closely related to distributed application development, any applicable methodology can be used. The best choices are those targeted at object development, where significant user input is required, and development timescales are short. Appropriate techniques include DSDM (Dynamic Systems Development Method), RAD (Rapid Application Development), JAD (Joint Application Design) and XP (eXtreme Programming).

The more successful methodologies used in web application development are those that have a strong focus on prototyping, along with the ability to cope with tight time-boxed development cycles.

4.2 Site flow

Once use cases have been defined they can be used to determine the overall site flow. Where user interactions are required, these can be mapped to screens, and inputs and outputs defined. This will allow the design team to determine the amount of dynamic content, along with user inputs. This process can be considered a variant of a traditional workflow development, as the site flow links application components together.

The site flow can also be used to indicate where control of the application is handed over to third-party services, such as credit card validation systems.

4.3 Object design

In conjunction with site flow, the use cases can be used to define the components and objects used in the web application. Standard object-oriented design methods can be used by the design team, with detailed design of object interfaces a key deliverable. As the Unified Modelling Language (UML) has become a popular means of handling this process, it is a likely candidate for the object design phase of a web application.

It is worthwhile using modelling tools to develop stub code directly. Tools like Rational Rose are important development aids at this stage of the development process, as they are able to take UML object descriptions and deliver outline code and interface definitions for most common object-oriented languages, including Java and C++.

5. DEVELOPMENT TOOLS AND ENVIRONMENTS

Once the design process is complete, it's time to start development. While initial prototypes may have been delivered during the design cycle, they are unlikely to be used within the development process – and will have been delivered on tools suitable for use in a workstation environment.

While the application and infrastructure architectures will have defined the overall deployment platform, development requires a mix of server tools and client applications. Critical features are debugging environments, platform integration and links to code versioning solutions. It is also useful to implement some form of community solution for the development team, so that they can exchange information in a structured environment, while building the application knowledge base.

5.1 Open source technologies (Linux, Apache, MySQL, Perl/PHP)

The open source movement has focused a large amount of its development effort on web applications. The Apache web server is the world's most popular server, and has developed a reputation as a stable work horse that can cope with most demands. Its extensible architecture has made it the centre point of what has become known as LAMPs – Linux, Apache, MySQL, and Perl/PHP.

A low cost alternative to the Microsoft and Java web application routes, LAMPs offers developers everything they need to create SME solutions. While enterprise development is possible, some of the underlying technologies do not scale as well as others. There are also issues around integration with load-balancing applications and appliances, and with legacy systems where Linux-specific drivers do not exist.

Developers wanting to work with LAMPs will need to learn how to develop CGI-based script applications using the popular Perl scripting language. This is a powerful tool that is designed to process text and lists of information – exactly the model used by web browsers and web servers. Alternatively PHP, an in-line scripting technology like Microsoft's ASP, can be used. Both Perl and PHP developers can take advantage of online libraries of components and examples that can be used to reduce development time.

The MySQL database may not have all the features of Oracle or SQL Server 2000, but it offers most of the basic functionality needed for a web application database. An alternative exists for larger scale applications in the shape of Postgres.

Development tools for LAMPs are few and far between, and most developers will stick with a standard text editor. NuSphere has developed a set of pre-configured LAMPs products, which it bundles with web-based administration tools and a PHP targeted development environment which also includes debugging facilities.

One of the most popular code versioning tools is CVS, the Concurrent Versions System. This is a powerful, network-centric tool that can be linked to issue and bug tracking tools, as well as offering an open framework for the development of clients. This has led to the development of CVS tools for most operating systems, including development environment integration.

5.2 Microsoft technologies

Microsoft has made significant investments in web application development, offering tools in the standard Office suite that allow rapid deployment of small database-driven applications, to large scale enterprise development through its Windows 2000 server platform and the Visual Studio development environment.

Key technologies are Microsoft's COM component architecture (and the Visual Basic and Visual C++ languages), the Transaction Server application server and the ASP (Active Server Pages) server-side scripting environment, managed through the Visual InterDev web application development tool. These allow developers with existing Microsoft skills to design, develop, and deploy web applications with minimal additional training. Microsoft's own web server, Internet Information Server, is bundled with the Windows 2000 operating system, and acts as a host for ASP web sites. Third party tools like Macromedia's Dreamweaver UltraDev can be used to develop ASP applications, and offer developers an alternate approach to working with these technologies.

Microsoft has also developed a selection of related products, including the Commerce Server 2000 store and workflow development tool, and the SharePoint document management portal. These tools can all be extended with custom COM developments, and integrated with other Microsoft server products. Other tools suitable for use in web applications include Application Centre 2000, which manages application package deployment, as well as offering improved load balancing features.

The Microsoft development platform also includes a source versioning tool, in the shape of Visual SourceSafe. This is integrated with the Visual Studio development suite.

The current generation of Microsoft development products is reaching the end of its life, and is due to be replaced with Visual Studio.NET. This introduces C#, a language targeted at web application component development, and an XML based web services architecture that allows web applications to be used as components in other applications.

5.3 Java technologies

One of the most common web application development platforms is Java. Its network-centric component model is ideal for developing large scale distributed applications, and a wide range of application server products means that complex component applications can be effectively managed.

Key Java technologies include Java Server Pages (JSP), which embed Java code in HTML to deliver dynamically generated content to web browsers. These can be linked to servlets, server-side Java components that act as a link between business logic and presentation layers of an application. In a small scale application, servlets can provide business logic services, along with access to data and legacy solutions. However, at an enterprise level, business logic is best handled within the management framework of an application server.

Application servers are designed to host Java application components, and provide tools for pooling data connections, as well as marshalling communications with web servers. When implementing Java 2.0 Enterprise Edition (and Enterprise Java Beans), application servers are available from many different vendors, including Sun, IBM and BEA. While some application servers like BEA's WebLogic concentrate on providing tools for managing EJB applications and component packages, others like ATG's Dynamo provide more in the way of libraries of tools that support user interface development.

There is a substantial body of expertise that has been built around Java web application development. Whether in the shape of on-line communities, or developers' libraries of design patterns, these expert resources can reduce risk and decrease development time.

Like Microsoft's .NET initiative, the Java world has also indicated that it will be offering tools for working with web services in the future. XML tools are available from the open source Apache Foundation, and web services ready application servers have been delivered by Iona and IBM. Sun has begun the process of defining ONE – the Open Network Environment – as its answer to Microsoft's .NET. This is intended to produce a common framework that will allow compliant application servers and Java code to work with remote objects exposed with web services.

Development tools for Java come in many shapes and sizes, and each Java developer is likely to have their own favourites. Popular tools include IBM's Visual Age for Java, Sun's Forte (and the closely related open source NetBeans tools) as well as traditional text editors.

5.4 Other technologies

While the bespoke approach to web application development is attractive, it is possible to work with packaged solutions to deliver effective results. These solutions range from proprietary development environments to specific applications.

One approach, where money is little or no object, is to take advantage of the large scale application frameworks offered by vendors like Vignette. These mix content management solutions with the features needed to deliver a full scale set of web applications. Some (often significant) development effort is required to customise the application framework and deliver the appropriate solutions.

Closely related, but more focused on specific functionality are tools like the Plumtree enterprise portal. These can be customised by experienced developers, but also offer considerable out-of-the-box functionality. Other tools are available for document management and collaboration.

Tools like Macromedia's Cold Fusion can be compared to application servers. However, instead of working with well known languages they offer their own proprietary solutions. Cold Fusion extends existing web server functionality with its own set of mark up tags that can be used to develop data-aware applications. Conditional programming techniques can be used, as well as some object-oriented functionality. While not as flexible as working with a true application server, this approach does allow for extremely rapid application development, and has been used to deliver complex heavily used web applications, as well as smaller quick-win internal solutions.

At the other end of the scale are the dedicated store development tools. These range from simple single catalogue, single store tools like Actinic Catalog to complex e-commerce platforms. Most tools in this category offer at least basic customisation, while the higher end tools offer APIs that can be used to add functionality through new components.

In some cases it is possible to take advantage of the work done by other organisations, using Application Service Providers (ASPs). These services can be integrated with your existing business processes, or branded and used within your own applications. This model is an important one, as it also points to how the next generation of web services are likely to operate.

6. BUILDING AND DEPLOYING WEB APPLICATIONS

The design process leads directly to development and deployment. This engineering phase is intended to take the application design and turn it into a fully functional web application – along with any supporting documentation.

6.1 Prototype often

Progress is best shown with tangible results, and the web application development team's watchword is 'deliver often'. While fully functional code drops may not be available every day, regular prototypes allow governance teams to see just how application development is progressing – as well as helping creative, business and technical development resources to work together.

An early deliverable will be a wire-frame walk through of the site. Lacking back-end business logic, it will provide basic page and site flow frameworks that will help the creative team develop appropriate look and feels, while allowing business analysts and the business logic team to understand how use cases will be implemented. The wire frame can then be modified as server side business logic is added.

Using this approach, prototypes can be released for usability testing as significant development milestones are reached. The loosely coupled component nature of web applications allows functionality to be added as new components are released. This will allow key functionality to be demonstrated as it is completed, and to help business analysts to determine that the functional requirements can be supported by the web application.

6.2 Usability studies

Prototyping is an important tool in developing the usability of your web application. Even a wire-frame site prototype, with few graphical assets included, is an important tool for testing and developing the information architecture of your site. Working with an information architect and a usability engineer, a development team will need to produce a usability test programme.

Using each prototype, selected end users should be given specific tasks to complete. Based on the initial use cases, these will be used to make sure that site flow and the navigation metaphors chosen allow users to interact with the application successfully. It is a good idea to video some users, in order to show the visual designers exactly how their designs and layouts are being used. This can be used to fine tune designs and site flows in order to deliver the most appropriate user experience.

Interviews can also be used to determine how end users react to the site design. It is possible to have an excellent, efficient information architecture combined with an unattractive choice of design assets and a disagreeable colour palette. While the site may lead users to make successful transactions, it must also attract them back in the future.

6.3 Development test frameworks

In any component architecture, component testing is an important part of the development process. As the component model demands that each component has well defined interfaces, it is possible to develop test frameworks that can be used to test components as they are developed, reducing integration risks. They should be able to offer the ability to show inputs and outputs, and to indicate whether responses are appropriate.

Before integration commences, all components should have passed basic functionality tests. Test frameworks can also be combined and modified to handle combinations of components. Where databases are tested, stand alone SQL queries are a suitable tool, and should be supported by most development environments.

Once components are tested and complete, test builds can be deployed on the development server environment.

6.4 Staging server

Once component development and package customisation is complete, builds can be deployed from development platforms to a site staging server. This is used as an internal QA platform, where builds can be tested before deployment to public facing environments. Dummy interfaces to third party systems can be used to ensure that an application is fully operational. The staged application should also operate using a snapshot of any databases used, in order to simulate a fully operational system. A content feed should also be provided from any CMS used.

The staging server can be used to test new builds prior to deployment, or to validate roll-backs if a deployed version of the application fails when exposed to live conditions.

6.5 Deployment and systems integration

The final stage of the development process before launch is the installation of the application on the final production environment. One option is to use a tool that can package the application from the staging server, and then deploy it directly onto the production system.

Once deployed, final integration testing can begin. This will allow an application to be connected to third party systems, as well as live data sources and content streams. When integration testing is complete, the application can be handed over for user acceptance testing.

6.6 User acceptance testing

Once the web application has been completed and deployed, it will need to be fully tested to show that it operates in accordance with the functional specifications. Where project scope has changed during the development process, the functional specifications will need to have been updated through change control procedures.

Test procedures will be defined using the functional specifications as a guideline. User acceptance tests will include full use case testing, as well as exception tests and site flow tests. A team of testers will be required to carry out these tests, and all results will need to be documented, as they may be required for contractual acceptance.

However a critical component of these final tests will be performance testing. A wide range of tools are available that can take a scripted approach to site testing. Using clusters of workstation class PCs, these tools can simulate large numbers of users – with typical usage profiles. These tests can be used to test site performance under load, stress testing the application. While stress testing product suites exist, they can be expensive, and lower cost open source alternatives are becoming available.

6.7 Soft launch

Once a web application has been tested and deployed, it is good practice to soft launch the application prior to a full release. In a soft launch a selected group of users is given access to the application, and its performance is monitored. This process allows any remaining bugs to be trapped and rectified before the site is launched.

The soft launch process also allows any operational processes to be bedded down, especially for system administration, content creation and call centre staff.

6.8 On going monitoring

The fully public web application will need to be regularly monitored. This will allow any errors and problems to be identified, along with requirements for future releases. An important tool is the web server log file, as this allows user behaviour to be captured and analysed. A good web log file analysis tool will identify user paths through a site, helping determine the percentage of users that complete a transaction, as well as the routes they use to access information. This data can then be used to modify navigation schemes in order to promote certain pages and application features. The log files will also indicate load problems and possible application errors.

7. ONGOING ACTIVITY

Once the first release of a web application is up and running, it is time to start planning the next release. Using the results collected from various monitoring tools and log files it's possible to get a clear view of how an application is being used. Together with changes in application scope, and the resulting new requirement specifications, the whole design and development process can be run again, producing release 2.0 of the application.

Web application development doesn't stop with the first release of a site. It is an ongoing process that needs to be responsive to user needs, and to changes in functional requirements over time. A large commercial application may still be under development for its whole life – just look at Amazon.com as it rolls out new features every few months. An iterative lifecycle is a challenging one for an organisation, but a well designed and well run web application will be an important part of a business – and one that will deliver a return on investment.

K E Y P O I N T S

- **Web application developments haven't stopped despite the end of the dot com boom.**
- **Requirements definition is a critically important part of web application development.**
- **Functional specifications need to be defined early in the development cycle, along with usability requirements.**
- **High level design starts with a system architecture, and then focuses in on the application's information architecture, application architecture, infrastructure architecture and security architecture.**
- **A release plan should be written before the application development begins in earnest.**
- **Development methodologies should be chosen that support short development cycles and object-oriented component development.**
- **A wide choice of development environments exists – from low cost open source solutions to enterprise-level Java systems built around application servers.**
- **A key engineering tool for web application developers is rapid prototyping.**
- **Prototypes can be used to confirm and test usability and page designs.**
- **Test at a component level, then at an assembly level before deploying to a staging server.**
- **User acceptance tests should include performance and load tests to ensure that the application is fit for purpose.**
- **Once live, an application should be monitored closely in order to aid plans for future releases.**