
Patterns of Experience

A Review of IBM's Patterns for e-business Initiative

Authors: R Bloor & M Hanrahan
Date: June 2001





Management Summary

Repeating Patterns

“Throughout its history, the computer industry has devoted an inordinate amount of effort to reinventing the wheel”.

Instant Summary

Patterns for e-business is a methodology developed by IBM with the aim of helping senior executives plan ‘e-business’ system architectures. The need for such a methodology is considerable because e-businesses have evolved to such a point of complexity it is almost impossible for a chief executive to keep up to date with all the essential features. Bloor Research considers the Patterns to be a significant and worthy initiative that delivers what it promises. We anticipate that the methodology demonstrated within the book is set to become essential for the modern IT professional.

Key Findings

This paper provides an overview of IBM’s Patterns initiative.

- The Internet is continuing to grow. Despite the ‘bursting of the dot-com bubble’, companies are continuing to generate profits using the web as a channel.
- Companies that make money from the web live and die by the efficiency of their system architecture. Today an e-business system must not only deliver 24x7x52 availability and scale to sudden traffic spikes of millions of users, it also must be capable of successfully integrating a diverse range of legacy architectures as a result of purchases and acquisitions. All this must be done with budgets that have been drastically reduced since the dot.com crash.
- According to the Standish Group’s 1995 Chaos Report, 31.1% of projects get cancelled before they’ve even begun. Due to the demands listed above this failure rate is probably much higher today.
- Software architects are rare, even amongst consultancies and system integrators.
- Although e-business systems are complex in nature and form, there is considerable re-use of functions and components within them.
- IBM’s Patterns is a clear and concise methodology that maps these re-usable functions and components. It uses variations on only six basic models.
- The Patterns for e-business methodology works in conjunction with IBM’s website to work as a practical toolbox for individuals at all levels in the organisation.

The Bottom Line

Senior executives with the responsibility for an e-business system are recommended to arm themselves with IBM’s Patterns. We found the Patterns to be a lucid methodology that significantly improves the reader’s ability to understand system architecture strengths and weaknesses, and also their ability to articulate them to both the technically fluent and the technically illiterate colleague.



The Risks and Rewards of e-business

After The Gold Rush

We have witnessed the swing of a mighty pendulum. At the height of its swing, the new economy boomed, fed by a flood of money from institutions, venture capitalists and small investors. Greed drove the market, as it always does in such times, with unrealistic expectations following in its wake. Then the pendulum began to retrace its path.

As it swung back, greed turned to fear and the flood of investment money dried up. The pain amongst the dot coms was palpable. There were cut backs, lay-offs and closures as the value of the once mighty dot coms plunged to the depths. Rumours even began to circulate about the prospects of some of the dot com stars like Amazon and Yahoo. The technical stocks were infected and so were the Telco stocks. The NASDAQ returned to where it was a few years ago, and then continued to fall. The gold rush was well and truly over.

Although the Internet gold rush stopped dead in its tracks, the Internet hasn't noticed. It continues to grow remorselessly, oblivious of the plight of investors. The Internet landscape may be littered with the wreckage of failed web businesses and initiatives, but it is also peppered with genuine Internet successes like eBay, Yahoo, AOL and others. It is also peppered with genuine bricks-and-mortar successes; companies that have learned to exploit the medium and its associated technology. Here we can point to Wal-Mart, Charles Schwab, Marshall Industries and many others. Outside the US, the picture is slightly different with fewer dot com successes and greater success from the old economy.

The problem that the early Internet pioneers had goes by the name of business infrastructure. With new ideas and their much-vaunted "first mover advantage", they were able to attract high levels of web traffic. But many had problems building the necessary infrastructure to deliver on their business plans. Service to customers was generally poor and sometimes abysmal. While the stalwarts of the old economy were late to the market, they had much of the required infrastructure in place, even if it wasn't ideal. They had outlets, distribution operations and brand equity.

The opportunities in the B2C (Business-to-Consumer) market were only a part of the picture. The B2B (Business-to-Business) market is actually much larger and it developed later. Here the dot coms were very weak indeed and made little impact. The early success stories came from the IT market with Cisco, IBM, Oracle, Dell and many others trimming substantial costs from their operations. The IT vendors were eating their own food and doing well on it. Others followed. However, while there were success stories, there are also many failure stories.

The e-business Challenge

The Internet introduced a whole new set of challenges and not all companies were able to deal with them. Even successful companies, such as Amazon, E*Trade, Charles Schwab and eBay, all of whom had ample funds for building computer systems suffered problems and outages - some of which became headline news in the business press and sent share prices into a spin.

Building usable web sites is not a simple task. It is worth absorbing the following facts:

- a) Web sites are on all the time; 24 hours per day for 365 days per year. Prior to the Internet there were very few computer systems that had to be "non-stop" and much of the technology that is used to run web sites is not "non-stop" either.
- b) Some web sites attract users in the millions. Prior to the Internet no computer systems ever attempted to cater for such large numbers. Even thousand-user systems were rare.



- c) Outages and performance problems are, understandably, seen as poor service by users. They used to accept it more readily than they do now. Expectations are getting higher.
- d) Most web site users cannot be trained to use the applications that they use. They will not recognize problems and may not report them at all.
- e) The investment boom that generously gave the dot coms funds to finance and recover from their IT errors is now over. The cost of errors has risen accordingly.
- f) The Internet is still growing, adding more people by the day. It will grow again with the proliferation of mobile devices and will grow further when embedded chips are added into the mix. The difficulties of e-business systems will multiply accordingly.

The On-going Record of Failure

Research produced by The Standish Group in its Chaos Reports has repeatedly shown that there is a very high rate of failure in large computer projects. In 1995 its US survey reported that **“31.1% of projects will be cancelled before they ever get completed”**. The costs of this are difficult to calculate accurately because the full cost is not just the project cost but also the opportunity cost. However the costs are high and when a mission critical project fails, the opportunity cost can be huge.

The research estimated that in 1995 US companies and government agencies would spend \$81 billion on cancelled software projects. This is aside from the fact that many other projects would be delivered late and exceed budget. It found that only 16.2% of projects were delivered on-time and on-budget. In considering these figures, one would do well to remember that in 1995 nobody built e-business systems that needed to run all the time and perform well with hundreds of thousands of users. In all probability the failure rates for e-business systems are far worse.

The Internet transformed the IT world. It became clear that the browser was the new user interface, superceding Windows. Most new systems that were built were built with this in mind and many old systems were refaced to allow usage through the browser. Companies did not just build web sites to interface to the world, but built “intranet” sites to provide interfaces to internal systems and “extranet” sites to provide special company-to-company capabilities. So the trend developed for all applications to adopt the same kind of interface and thus become “e-business” applications of a kind.

There are many reasons for project failure, but poor architecture is a major one, especially in the e-business arena. Software is deceptive to the eye. Large software applications are never simple to build, but to the user they look no different to small ones. However, large scalable computer systems are not easy to build. They require experts, they take time and they need to be designed correctly. This is particularly true of e-business systems.

It should be no surprise then that many e-business projects failed. Sometimes it was because the business idea was flawed, but more commonly it was because the **architecture** was insufficient for the task in hand. For those who are less familiar with computer systems, it may seem strange to talk about architecture - but the word is well chosen. Big systems need an architect - they cannot just be thrown together and expected to work.



Dancing Around Architecture.

The Architecture Problem

There is a shortage of skilled software architects with sufficient experience to design large systems. Those that have successfully built e-business systems are small in number and much sought after. Given the importance of such systems one would think that the IT industry would somewhere provide adequate guidelines and best practice advice to assist. However such information is quite hard to find. Information technology changes so quickly that professional bodies and information resources have not developed in the way they have in the medical world or the engineering world. Knowledge and expertise exists in consultancies and systems integrators that specialize in building large systems – although even there it is rarely formalized and available for project staff to consult.

The skills shortage in the IT industry is extreme and experienced software architects are rare. Architectural skills are rarely taught and usually acquired the hard way - by doing a project badly and discovering what was done wrong at the review stage. And yet, software engineering is not so different from other engineering disciplines.

Admittedly the technology changes rapidly, but this is a double-edged sword. The year-on-year increase in computer speed, for example, makes life easier in the area of performance, while the regular appearance of new types of application makes life a little harder.

Nevertheless, the basic architectural facts remain roughly the same. When an application runs, data has to be brought from permanent storage into memory to be processed and then stored for later usage when the processing has completed. Processing power is required for presenting information to the user and for manipulating the data and for getting it and putting it away. There are many complications in these activities, but these facts have been true since the advent of on-line computing in the 1970s and they are still true now.

In order for an application to perform well, the processing activity has to be distributed across the available resources (terminals, PCs, networks, servers, etc.) and an architectural plan for running the application(s) involved has to be created. Because the basic facts change very little, one should expect the number of practical workable solutions for deploying software over a network to be few.

This is true in every other area of engineering. Take for example, the architecture of buildings. This was first documented in the Renaissance with the publication of Vitruvius' 'Ten books on architecture', and as the record demonstrates the structural engineering choices were few. Even though architecture has evolved dramatically over the years the fact still remains that the options are few even though the final products may display great variety. We should expect the same to be true of software engineering and, as it happens, it is.

Research done by IBM indicates that there are only a few feasible architectures for large systems and particularly for e-business systems.

Patterns In IT Architecture

The fact that architectural choices in computer systems are limited was never a secret. Competent software engineers in large computer sites know this, systems integrators, consultancies and computer companies know this. However until IBM began work on analyzing and defining the full range of possibilities, this knowledge was never formalized.

There are reasons for this: Large computer systems are never written from scratch. They usually involve the use of a very variable set of components; different servers, possibly from a



spread of vendors, different operating systems, different arrangements of the computer network, different database products, different middleware products, different front-ends (including varieties of PCs as well as other devices). In addition to this the software development products employed may vary, working in slightly different ways and the system may also have to interact with other existing systems or software packages, either by passing data or even exploiting some of their capability. Finally, there is the fact that new software components with useful capabilities appear on a regular basis.

Thus, it is rare that any two large systems are exactly the same or even similar at the level of the components used. A simple way to look at this is to consider an end-to-end transaction, say the placing of a purchase order. From the user perspective, all that needs to happen is that the application works in a coherent way, responds quickly when the user is at the keyboard, and rarely fails. If that is satisfied then orders can be processed. The service that the user requires is easy to define.

The job of the architect is to design the system to satisfy this requirement for a given population of users. While systems that cater for a small number of users have a lesser requirement, an architectural design is still required for them, if there is any chance that they may grow. The architect may have the ability to select some components, but may also have to work with the components that are dictated by history or policy. He will also have to cater for security, system recovery, system management of the facility and other issues. And if the number of users is likely to increase with time, as is typical with e-business systems, then the system must be able to scale to accommodate the growth of usage.

Thus although there are few architectural choices for deploying the system in a corporate network, there are many complications.

The problem that IBM is attempting to solve with its Patterns initiative is the problem of reuse. Although the same problems repeatedly arise in many areas of building software systems, the details of how to solve these repeating problems is not widely shared, because it is rarely formalized. The problem of insufficient reuse exists in many areas of software and the area of IT architecture is no exception. This is what IBM is seeking to resolve.

IBM's work on "Patterns for e-business" began in its Global Industries Group and Software Group as "a major IBM initiative to establish a standard ". It has since grown in sophistication and detail, and continues to grow.

Put simply, the idea is to provide a knowledge resource that can guide IT staff through the architectural choices available to them. It is intended to cover:

- a) The business process the system must deliver.
- b) The business and IT drivers that affect the choice of both process and solution.
- c) The application patterns and their logical components that are best suited to delivering the process.
- d) The runtime patterns and actual technology required to make the application pattern work.

The provision of such a knowledge resource requires the assembling and formalising of a large body of practical experience and renewing it on a regular basis so that it remains relevant and current at all times.

The patterns boil down to a set of six high level patterns that can be used and re-used to create systems unlimited in scale. They consist of four primary Business patterns and two Integration patterns. They are summarized in the table on the following page:



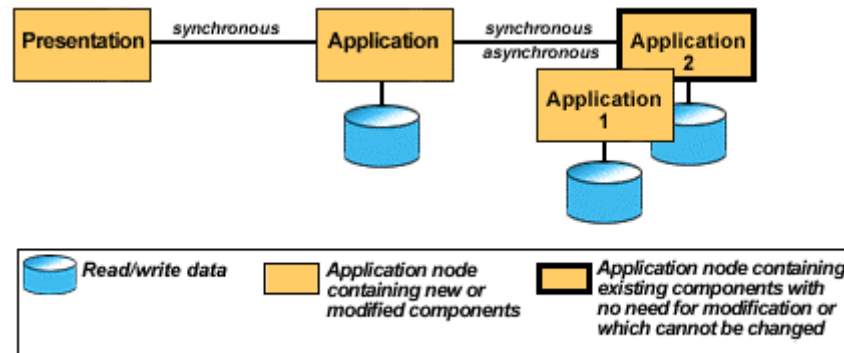
Business Patterns		Description	Examples
1.	Self-Service (User to Business)	Applications where users interact with a business via the Internet	Simple Web site applications
2.	Information Aggregation (User to Data)	Applications where users can extract useful information from large volumes of data, text, images, etc.	Business intelligence, Knowledge management, Web crawlers
3.	Collaboration (User to User)	Applications where the internet supports collaborative work between users	Email, community, chat, video conferencing, etc.
4.	Extended Enterprise (Business to Business)	Applications that link two or more business processes across separate enterprises	EDI, Supply chain management, etc.
Integration patterns			
5.	Access Integration	Integration of a number of services through a common entry point	Portals
6.	Application Integration	Integration of multiple applications and data sources without the user directly invoking them	e-Marketplaces, eCommerce, etc

This table illustrates the four primary Business patterns and the two Integration patterns that can be used to combine them. Business patterns can be defined in terms of the interactions they involve between users, applications and data. A user can interact directly with an application (User-to-Business) or interact primarily with a data resource (User-to-Data). Applications can interact between businesses (Business-to-Business) or users can directly interact (User-to-User). Each Business pattern further breaks down into Application patterns, which further devolve into the Runtime patterns detailing the technical solutions that implement the Application pattern.

The major strength of the Business and Integration patterns is the way they can be used as the basic building blocks of very complex systems. Combining these patterns together in a bespoke way results in a custom design specific to a particular customer requirement. On the other hand if this same combination of patterns is found frequently this is referred to as a Composite pattern. An example of the latter is those e-Commerce solutions that are based on the Self-Service business pattern and the Information Aggregation business pattern (based on the need to aggregate the catalogue from multiple existing sources) integrated by the Application Integration pattern. However, a modern e-Commerce system will not be limited to just a user accessing an online catalogue and making an order. It will need to link up to a credit checking service (Extended Enterprise), contain transaction processing, credit checking, access to the companies ERP system and ideally should record everything in the companies CRM system. This combination of Business and Integration patterns is called the e-Commerce composite pattern. Understanding these six patterns will give a CIO the ability to extend and distil all the required functions of the complex, rapidly evolving system architectures demanded by the business market.

Choosing and Using Patterns

Once it has been decided which pattern is appropriate then the software architect can examine the various computer configurations that IBM has documented **and which are known to be workable**. For each Business pattern IBM has documented the appropriate Application patterns. An example of a User-to-Business possibility is shown below:



Without going into technical detail, we can point out that there are choices at this level and this is one of the choices. The motivation for selecting this rather than another will depend on context. For example, a business may choose this pattern because it already has some specific computer hardware in place and because this pattern represents the least cost alternative.

Once such a selection is made and checked for appropriateness (how scalable is this configuration for example) then it is possible to move to a lower level and select the appropriate software components to support the application. Here decisions need to be made about databases, security, systems management, directory services and many other components that are required for the system to perform, be robust and be manageable. Again, many of these decisions emerge from the context of the existing IT configurations that are already deployed. If there is already a particular database product in use and it can provide the required level of performance, then it will select itself. Similarly decisions may already have been made in the area of operating systems, system management and security. The new system will inherit these.

IBM Patterns help by flagging those product combinations that have previously been used successfully in this kind of system and if there is no information on this (as for example when a totally new software product is to be used) the patterns documentation will alert the architect to this fact – which may then mandate some preliminary product testing before a decision is made.

To summarise the use of Patterns, the process begins at a high level with the architect selecting the type of system that is being built and “drilling down” into the detail, gradually making decisions as the architectural possibilities are examined. Once a basic approach is finalized then product choices can be examined. Once done, the architect can review the guidelines that are provided which tell you how to design, develop and manage the application. The process is not a simple one because at each stage there are issues of cost and time to consider as well as the robustness, security and scalability of the system being built. However at each stage a set of practical information is made available to the architect to assist in the decision making process.

The outcome of this process is that the systems architect will be in a position to know which choices are risk-free (because they have been proven before) and which choices have some risk attached. These may need to be proved by the product provider. Finally there is also the part of the system that is being developed from scratch. Here the architect will be appraised of



the architectural requirement of the other software products that the new application must integrate with and may also be assisted in selecting the appropriate development software.



The Evolutionary Knowledge Resource

Back to the Future

We referred earlier to Vitruvius' books on Masonic architecture, which were published first in 1511 in Venice and republished repeatedly. They provide a parallel to IBM's effort in the area of e-business Patterns. These books were a comprehensive formulation of the knowledge of the day, which had hitherto been closely guarded secrets amongst Masonic guilds. A new information technology – printing – had begun to proliferate and the knowledge was shared.

The CEO in charge of today's company stands in a similar position to the Renaissance prince who wanted to patronize a church but was a little wary of the state of his coffers and the politics of the renaissance "consultants and engineers" he would have to employ. However with Vitruvius' tomes he was able to stroll comfortably about the site and ask intelligent questions – for example, about the quality and strength of the materials in the critical supports. Armed with an underlying knowledge of the principles of the trade and the possible choices that could be made, he had a better chance of being listened to and respected.

CEO's, CFOs and CIOs in charge of constructing e-business systems face a similar degree of complexity and gravity of investment. IBM's work on Patterns provides an entry point for such individuals to acquire a closer understanding of the engineering possibilities and risks. It enables better decision making.

The Problem of Evolutionary Documentation

The evolution of the architecture of buildings was relatively static over many years and thus its encapsulation in the printed word was an effective means of dissemination. IBM too has chosen to go into print with the work it has done on Patterns, publishing a book, (Title: Patterns for e-business: A Strategy for Reuse, Authors: Jonathan Adams, Srinivas Koushik, Guru Vasudeva, and George Galambos. Published by IBM Press, ISBN: 1-931182-02-7) which explains the principles of architectural patterns in depth.

However, IBM recognizes that this is not enough. The pace of change in the IT industry is so fast that books can be out of date before they leave the printer. In a realm where the details are so critical and yet change at such a rate, this presents serious problems. A book is ideal for teaching the over-arching patterns and principles of e-business architecture, but is unsuitable to the changing complexities of the working parts of e-business systems.

IBM's solution to this problem is to provide an evolutionary resource - a Web site that acts as an updateable repository of detailed information. However, while the Web is the medium IBM has exploited to provide its reference information, the really impressive aspect of the Patterns web site is its intelligent use of hypertext. It may be obvious to the casual observer that a Web site, with its links and hot spots, is a suitable platform for hypertext but few Web sites exploit this to the full. Most companies only use it as a distribution mechanism for press releases, white papers and FAQs.

The potentially infinite and incremental nature of hypertext has hardly been exploited at all by web sites. Hypertext significantly improves on the linear form of a book, enabling readers to choose their own path through relevant information in real time, while still remaining in context. ***It is this – IBM's Web site - that transforms the Patterns for e-business from being an interesting information resource to a practical tool box for individuals at many levels.***

IBM's use of hypertext mixes pertinent business (e.g. what kinds of applications in what industries have been built using a given set of choices) with important technical information (e.g.



which products were used, what problems could be encountered, how to avoid such problems etc.). In other words the resource is relevant and useable at many levels.

For example, the value of a high-profile prestige project like the Sydney Olympics is far greater than the sum of the kudos gathered by its advertising. This project was one of the largest stress testing experiments the industry has seen, and the accumulated knowledge is of value to anyone that wants to understand difficult issues like scalability and performance – of far more value than any benchmarking.

An engineer interested in the scalability of a software component might not take the time to track down and study a thick white paper detailing how the Sydney Olympics website was constructed and run. However, if the information is presented in context as a menu choice attached to an explanatory diagram it may be a different matter. Similarly a business executive interested in building a very large e-retail system would be keen to know what level of specific retail transactions were achieved by the retail component of this and other systems.

Thus IBM's use of hypertext facilitates the sharing of business, architectural and technical experience and skills in a fluid, rapidly changing environment. It is a permanently evolving professional resource, but it is also educational – in the professional sense.

It can help to educate or re-educate the CTO that is no longer “up with the technology”, enabling him or her to comprehend the complexities of modern e-business systems. The patterns provide a practical framework to explore the technical details of these systems and drill vertically down as far as is needed.

The web site is incrementally added to, using customer feedback, reports from R&D labs and technical details of major projects that IBM, its partners and competitors (if they so choose) have completed. Where available FAQs, architectural options, technical manuals, stress testing results are placed in a business and technical context, using diagrammatic representations that are easy to understand and navigate through. The CTO can simply pick his architectural pattern, delegate the relevant sections to his engineering staff, browses the resources he thinks he should know – forwarding the rest to the engineers. The evolutionary nature of this knowledge resource means that anything of interest and value can be added to the mix – right down to libraries of applets or code components.

The Question of Independence

IBM provides the Patterns for e-business resource freely to anyone that wishes to use it and invests heavily in keeping it current. This naturally raises questions about the independence of the resource. As we understand it, IBM's view is that such a resource simply could not work if it were vendor specific, as very few sites (especially large ones) keep to a single supplier policy or even a coherent product strategy. Also, those that do will make architectural choices that they understand reasonably well. Their need for the Patterns resource is less.

What IBM is doing here is analogous to the “Linux/Open Source” idea, with its information resource being built up and maintained by its users which include IBM Global Services, but is open to anyone – other Systems Integrators and service providers, other technology vendors any customer, whether they buy from IBM or not. The advantage this confers on IBM is that it can ensure that the options that relate to IBM products and their capability are clearly available for review. IBM's Patterns resource will assist its own sales, but also that of Systems Integrators and service providers that may choose IBM technology. It may encourage Software vendors to interface to IBM products. It may encourage customers to choose IBM products and it promotes the IBM brand.



The skills and costs of building and maintaining the Patterns for e-business are considerable so IBM's competitors will need to make their own investments to instantiate the logical patterns using their own products. Given the openness with which the Business, Application and Runtime patterns have been published, competitors can use this information resource just as actively as IBM.



The Pay Off

The two obvious pay-offs that Patterns deliver are:

- **Risk Reduction:** Reducing the risk of making poor architectural decisions and poor product choices.
- **Skills:** Educating consultants and IT staff in the realities and capabilities of designs and products of which they have little or no experience.

However there are other benefits that are worth discussing.

- **Quality Control:** This resource can be used to assist in the quality control of projects that have been contracted out to consultancies and systems integrators. It could be used to frame intelligent questions to put to contractors and may even expose poor decisions.
- **Standardisation:** The existence of this resource encourages standardization as it can exposes situations where lack of standardization causes problems. It also exposes information on where standards are actually being adopted effectively (as IT standards are sometimes adopted only by the payment of “lip service”).
- **Productivity:** You need to explore the Patterns web site using a real example to get a true understanding of this. However our conclusion is that the relevance of the information and the intelligent organization of it, speeds up architectural work dramatically, reducing the time taken even by experienced architects in arriving at viable solutions.
- **Technology Sharing:** The Patterns web site has been built to allow specific technology sharing. Thus software vendors could use it to provide industry specific software components that are too specific to be worth turning into saleable product. Similarly technology users could use the resource to share software modules and even program code that have a specific context.



For Example...

Let us now briefly consider an example of how Patterns could be used in the context of a business:

Imagine that the CIO of an insurance firm has the responsibility for developing a new innovative 'future-proof' Web channel for his company to sell car insurance. Previously the company's prime channel in this area has been a telephone sales system, with back-end policy processing based on Unix computers and a batch-based billing system on a mainframe. His department has been given a budget but, from experience, the CIO knows it is tight and less than would have been provided in the heady days of the dot com boom.

The CIO has cut his teeth primarily in the realm of implementing and managing call centres where all the hardware and software technology provided was by suppliers. Privately he does not regard himself as an expert on system building. Nevertheless he has a keen sense of the potential of the Internet to increase the efficiency of this chosen line of business, and has confidence in his small team of IT staff. His regular scanning of the IT press, however, has made him wary of the 'future proofing' requirement in his initial brief.

According to what he has read, the world of the in-car PC and telematic service are only a year or two away. Despite all the uncertainty and hand-wringing over the future of the 3G networks, he knows that the localisation services built into 3G networks will open up new value added services to their customers. The CIO and his team have already done a considerable amount of brain-storming on just how a car insurance company might use an in-car PC and applications that use localisation services. Although they have come up with a few, including a localised map alerting customers they are about to park in a car theft black-spot, he knows that Return On Investment may not look good on paper. The relevant 3G wireless infrastructure needs to be in place and even then benefit to the customer is not a foregone conclusion. Despite this, he knows it would be commercial negligence to ignore any new or emerging business channels.

The more he looks at the brief and the budget, the more conservative it becomes. Hiring external consultants is simply out of the question – and anyway his experience in this area has been mixed. Let us imagine then that our CIO has studied the patterns book and, armed with a variety of mental tools, is confident he has distilled all the strategic necessities of the solution into a model of manageable components.

His primary business driver is to establish a new customer channel as soon as possible - classed in 'Patterns' as 'time-to-market'. Looking at the budget and knowing the business as he does, he resolves to establish a web channel that complements rather than competes with the telephone sales channel. It will provide read-only access to policy information, contact details, marketing and sales literature over the web. It could also provide business transactions that use simple data-base applications to give customers the chance to estimate what their premiums would be according to type of car, area they live etc.

Although full CRM capability would no doubt be desirable in the future it would require a level of back-end integration that could be added later. For the moment, the prime concern is to develop a solution that makes sure the company reaches the widest range of devices possible, and which can be quickly and cheaply customised to anything the "in-car PC" might throw at them in the future.

The primary IT Drivers of the solution are as follows.

- It must minimise the Total Cost of Ownership (TCO) of maintaining and building code, support and upgrades.
- It must exploit existing in-house skill-sets for implementing new technologies and integrating with legacy systems



- It should, whenever possible, chose re-usable, off the shelf components above custom code that is costly to maintain.
- It must also have 24X7 availability and complete scalability in hardware and software.

With these criteria established, the CIO has the choice of four primary Business Patterns for his solution. Although it is immediately obvious that the Self-Service model is the snugest fit - there is even an example of an insurance company given to illustrate the model, further investigation seems sensible in order to consider all options.

For solid 'Future-proofing', the CIO might want to investigate the other four patterns of information aggregation, collaboration or extended enterprise models and see what they subsequently add, both in terms of value and cost. If he thinks there is a definite potential roadmap for the future here, he might want to investigate further the Application Integration patterns that connect multiple Business patterns in process focused or data focused workflows.

Finally deciding on the Self-Service or User-to-Business model, the CIO looks up the relevant Application patterns. There are seven patterns to choose from, ranging in complexity from the Stand-Alone Single Channel application pattern, (the choice he finally makes) to the Agent application pattern, a complex agent-based solution suitable for personalisation and cross-selling to customers from a dynamic work In progress database.

The Stand-Alone Single Channel Application pattern aims to separate presentation from business logic through a web application server. Browser-based thin clients access the presentation logic of the server using the bare minimum of code (perhaps some Javascript). The presentation logic then accesses the business logic that is run wholly on the server. Keeping the presentation and business layers separate means the application can be quickly and swiftly custom configured to any kind of browser-based interface that may emerge in the future. Keeping the business logic on the server means that scalability simply requires the server to be up-graded vertically (scale-up) or horizontally by adding parallel servers (scale-out).

Confident he has a documented and tested Application pattern that meets all the needs, the CIO photocopies the model and related literature from the book, with commentary, and passes it on to his MIS (Manager Information Systems). He takes care to underline the importance of maintaining the separate nature of the layers and avoiding mixing scripting and components, as recommended by the Considerations section of the Application pattern page. His investigatory role is now over and the project is delegated down through his organization.

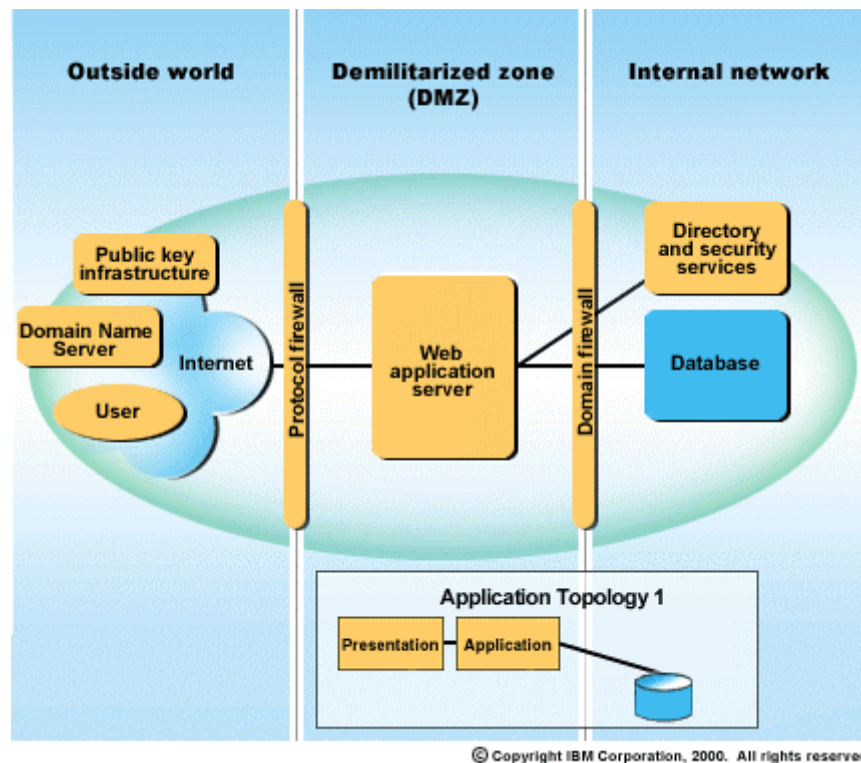
The MIS studies the model, which is now attached to the project brief, deliberating on it until he understands the implications. Then he visits the IBM Patterns web site to explore the detail further. He ends up looking at the User-to-Business Application pattern¹ page on the IBM Patterns website². (see diagram on the next page). His aim is to establish what the Patterns book refers to as a Runtime pattern.

A Runtime pattern is the logical architecture that is required to run an Application pattern. It is best described as the sinews placed on the bones of the architecture model that actually allow it to function. It embraces the middleware, the interfaces between nodes and the performance characteristics of the chosen components.

The MIS soon finds the basic pattern and notes that there are three runtime variations of it, which we can refer to as Variation 1, 2 and 3.

¹ The web site terminology is currently being synchronized with the revised terminology used in the new Patterns book. For example all references to application and runtime topologies will be replaced by application and runtime patterns. The User to Business terminology will be replaced by Self Service.

² <http://www-106.ibm.com/developerworks/patterns/u2b/at1-runtime.html>



The basic runtime description breaks the pattern down into a number of components. As illustrated, users access the web server through a protocol firewall using PKI and DNS infrastructure. The web application server, based in a DMZ (Demilitarised Zone), accesses the relevant in-house directory service and the application database through a domain firewall.

The second variation, Variation 2, adds a load balancer at the protocol firewall, and a shared file system to distribute the information required by the web application server. Both these implementations are proven and documented, and a list of guidelines and product resources is a single click away from the MIS in the sidebar on the left hand side of the page.

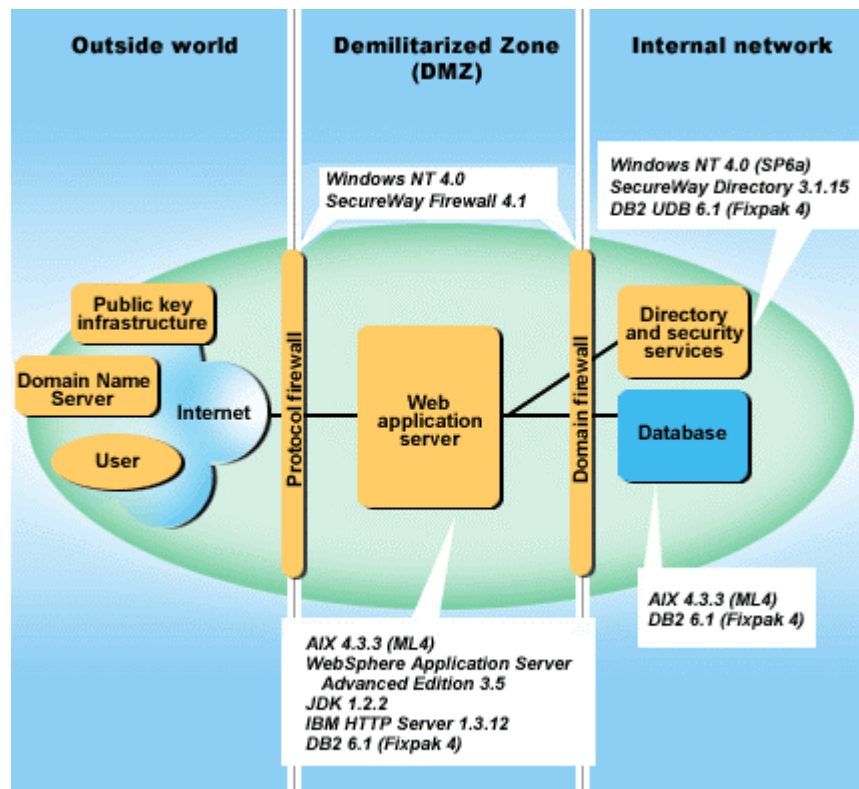
Variation 2 involves dividing the web application server into a web redirector inside the DMZ and a separate application server inside the domain firewall. The solution should increase the level of security.

Variation 3 is classed as an emerging pattern, but is clearly the one of greatest interest to our car insurance company, as it contains a pattern for enabling wireless access through 'pervasive' devices.

The model documents the essential new components for this solution. It flags the potentially problematic issue of who is going to have responsibility for configuring the protocol firewall, which in the case of our car insurance company, will probably belong to the cellular network providers.

Having seen the runtime variations, it is now the MIS's job to select the products that are capable of doing the job. The next web page¹ recommends that selections are made according to existing systems, platforms and skill-sets within the company and also, of course, scale effectively.

¹ <http://www.106.ibm.com/developerworks/patterns/u2b/at1-product-map.html> in this example



©Copyright IBM Corporation, 2000. All rights reserved.

There are five choices of platform shown at the bottom of the page: AIX, Windows NT, Linux, OS/400 and OS/390. Clicking on any will bring the MIS to a detailed list of products that are involved in implementing the pattern. If we consider the AIX option, then the diagram of product mappings shown above¹ appears.

This shows the exact interaction of all the components, and acts as a single access point for further information on technical manuals, recommended configuration and relevant case studies. Even if the company is a heavy user of Solaris (which might be an alternative to AIX, but for which product mappings are not currently available), the diagram is still helpful as it indicates what is required even if the product choices are not shown.

The MIS can now hand the diagram down the chain to the project manager and staff, who can be allocated responsibility for the chosen sectors of the model and asked to block in the current resources they are working with. The model can even be taken to product vendors with the request that they map exactly how and what they would recommend for the implementation. Results from multiple vendors can be compared before any decision is taken.

As this example illustrates, the Patterns provide a road map (and a fast route) through the architectural process, which can feasibly be pursued at a high level. Product tests can be organized according to the advice that is given. Finally when the project is complete the CIO might wish to provide information back to the Pattern web site describing the problems that were encountered in the project and how they were dealt with.

¹ <http://www-106.ibm.com/developerworks/patterns/u2b/at1-product-aix.html>



A Customer's Tale

Case Study: WaveBend

WaveBend Solutions, LLC, was founded in 1995 and describes its core business as focusing on 'providing dynamic and scalable e-solutions to fast-growing businesses, middle-market organizations and divisions of Fortune 1000 companies'. The company has more than 130 skilled professionals who have expertise in delivering complex technology to manufacturing, financial services, retailing, and high-tech industries.

WaveBend has been an early adopter of the IBM Patterns methodology, which they have been using since January 2000. The company's consultants use a number of process related tools when working with their clients, including Holosofx business process modeling tools that integrate with the Rational Unified Process, and MQSeries applications (for mapping XML across product solutions), but considers IBM Patterns to be central to both their sales and delivery process.

Ronny O. Neira, a Manager at WaveBend Solutions, stated that WaveBend uses the Holosofx software and Rational Unified Process methodologies to understand, document, and model the client's business requirements. The Patterns is used to develop the foundation and the high-level blueprint of the architecture. Once this is established, MQSeries software can be used to model the walls and floors in greater detail.

According to Neira, Patterns benefits WaveBend in several significant ways. Before adopting the Patterns methodology, consultants relied on schema developed in-house and consequently projects had a tendency to bottleneck around the systems architect.

'When we began, the skills of the system architect were at a premium, and it is still the case that investing in an individual's skills to the point where they become a certified system architect is an expensive process. The demands of the market are such that more e-business solutions have to be built by multi-disciplinary teams. IBM Patterns helped us, in a short space of time, to complement each team members' skill-set with an over-riding architectural knowledge. This architectural knowledge was of a level and consistency that enabled them to collaborate much more effectively with one another.'

Typically, consulting projects that involved developing architectural plans took 3-6 weeks to produce a full design. Neira estimated the Patterns methodology reduced the time by a factor of 30-40%.

'The theory was laid out in an accessible style that could be understood by all the workers on the project. In practice this meant that the highly qualified systems architect could be freed up to work on multiple projects at once, without the need to continually support his chosen solution with the different teams.'

The key strength of Patterns, in addition to this lucidity, was the breadth and quality of IBM experience on which the methodology was based.

'All of us are capable of making mistakes, but mistakes in this particular area of the industry can be costly. IBM's Patterns has been composed from the collective experience of thousands of architects and, from the word go, we were entirely confident that there was already significant risk reduction gained from taking on board this particular collective solution. We couldn't think of any organisation capable of that level of collective experience in the area in which we were specialising. We are very aware of the value of the pedigree of the knowledge we lean upon.'

The depth of the pedigree contained within Patterns for e-business is particularly important when developing a technology roadmap for the future. Strategies can be extended or changed



and calculated at will with the comfort of knowing that nearly every permutation and combination of the components will have already been successfully implemented somewhere by IBM.

The Patterns book states itself that the six basic patterns should supply 80-90% of the solutions needed in the business world. Neira has found that occasionally a solution fell outside of the range, but insisted the solution usually involved some 'tinkering with the code under the bonnet. The Pattern chosen still did its job.'

Conclusion

There is no room for idle dreaming and whimsical spires in the new age of system design. IBM's Patterns for e-business may be deceptively simple and nimble in form, but it should not be forgotten that the methodology carries a collective and silent legacy of some very serious investment and some very late nights.

IBM has made this legacy available for the price of a hardback book and a (free) Web site. This fact alone should guarantee 'Patterns for e-business' has a space on the bookshelf of anyone with the remotest interest in system architecture.