Table of Contents for Chapter 6

Part 1. Architecture 101 – 'The Language We Speak'

<<< … >>>

Chapter 6. *Architectura Sapiens* - 'evolution of one solution'

Homo Sapiens (i.e. us, mankind) have got to this point through the process of long and gradual evolution. *Sapiens* means 'wise', 'intelligent'. Some of us possibly are.
Architecture design of the complex computer system for the enterprise also will likely evolve through the levels of abstraction and more detailed design of various components.

We follow the Architect's train of thought in building the architecture for the specific enterprise project. We provide considerations for progressing the architecture design further, through several iterations of the design. This will give us an opportunity to demonstrate application of Quality Measures in the Architect's real-life scenario. We shall see how inter-dependant these Quality Measures are.

You will get a good feel for the dynamics of the design and what makes the large Enterprise Architecture so special. Hopefully, you will appreciate what is involved in getting there from the more simple architectures.

In this chapter, we mention specific products and vendors for the sole purpose of demonstrating the complete solution to the problem at hand. No attempt has been made to provide an inventory of all players in the field and their comparison. In your case, you may have different requirements and arrive to different decisions in selecting the technologies, products and vendors.

Business Problem

Let's imagine you are engaged as an Enterprise Architect in a sizeable project of building the online Internet-based application for the large Enterprise.

Enterprise provides services to a large customer base of users. As a part of its core business, Enterprise collects and handles large volumes of data on user profile and user transactions.
(Note - we intentionally avoid specifics of the data content in order to emphasise generality of the problem and solution. No doubt, you will be able to map the problem presented here onto business model of your Enterprise.)
Why not leverage this wealth of information assets by providing the attractive value-added services to a customer and generating a good business in the process, with the revenue stream and profits?

By assisting the customer in presenting and analysing his or hers own transaction data, we help the customer to make some important decisions in his personal spending patterns, recreational endeavours or in conducting their small business better.

Enterprise sees good opportunities in providing an easy access to customer data. Enterprise should be able to generate statistical and demographic reports, and present them as tables, graphs, bar charts, pie charts, as well as geographical maps – online over the Internet.
Marketing people and business analysts envisage that a lot of customers will be willing to pay for this service, either on subscription or usage basis.

Customer's personal data are very sensitive and confidential. Enterprise not only intends, but also is obliged and liable to keep it this way. Security and privacy of data access are of paramount importance.

Customers may be very proficient people in conducting their own business or recreational planning. However, we cannot count on a customer to be a computer nerd or Internet surfing addict (although,

chances are, some of them will be). User GUI interface should be self-explanatory so that customers able to navigate around the application and find the required queries easily.

In all likelihood, customer has a PC with Internet access at home, at work or at the location(s) of his small business, or all of the above.
Enterprise intends to provide customer with easy access to new service from any available to customer browser – PC at work during business hours, at home after hours when kids are in bed, in the public library on weekend, at the Internet café on occasion etc.

Customers will require good response times online over the usual low-bandwidths connection. Some customers will have ADSL or cable connections, at least on some of PCs they are going to use. However, most of subscribers will rely on a telephone line with a modem.

Enterprise expects a take-up of hundreds of thousands subscribers to this new service, and many of subscribers may become heavy users – logon often and run a lot of queries.

Customers may wish to access service at any time, from any place in the state, country, globe (space travellers may decide to check on their records too, when they have nothing else to do up there).

Service usage patterns will be somewhat uneven. This application may be in more demand at some hours of the day, some days of the week or months, some seasons of the year.

Enterprise has some existing IT deployment and online presence. New service needs to align its platform with existing infrastructure and technology guidelines.

And, of course, this application had to be deployed yesterday, on a shoestring budget… Just kidding, this will never happen to you.

Where do we start?

In the previous section, while describing the business problem, we defined an idea with a business model and high-level requirements for the new application.
We have received a first cut of both functional and non-functional requirements. We defined main features and the scope of the application. We have all reasons to believe that, by articulating the problem clearly in plain English, technical architects established rapport and reached mutual understanding with the business owners of the idea.

Your Enterprise will have some methodology or set of guidelines in place, with defined processes, procedures and deliverables that will map out the course of actions in building and deploying the application. We discuss methodologies later in a book.

This chapter focuses on technical and technological aspects of the solution to the business problem, on devising and refinement of the technical architecture.

Enterprise Architect comes to the party armed with the knowledge of existing guidelines and technical capabilities of the Enterprise, and with technical expertise in the information technology.
Enterprise Architect will look at the opportunities for re-use and smooth integration of components, servers and network connections, software, skills.

Rarely Enterprise Architects have a luxury of building the application from scratch. On one hand, we are fortunate to be able to leverage some pre-cooked ingredients of the architecture. On other hand, integration is hard on many levels, and exactly is the type of challenge where Enterprise Architect has to step in.
Solution shall achieve a degree of cohesion with the overall infrastructure, seemingly outside the scope of this particular application.

The more we can re-use or leverage, the lower the total cost of ownership (TCO), the shorter the time to market, the higher the overall quality of the solution, the happier the satisfied customer.

Business requirements have to be reviewed so that they are fully qualified and measurable, and quantified where possible. You may find that some requirements are not practical or viable, and, possibly, not really mandated by the true business imperatives.
You have to go back to business with this and clarify, resolve, find practical compromise, manage the expectations, explain the implications of unreasonably high demands (eg. on capacity and availability, with direct impact on complexity and costs).

Table below summarizes major high-level requirements for the business problem, with attempt to quantify or qualify them better for the design, capacity planning and scoping of the solution.

High-Level Requirement	*Qualified/Quantified Requirement*
Internet access	Thin client – browser compatible to MS Internet Explorer or Netscape version 4.0 or later
Bandwidth	Most of the customers will have an average PC on their premises with the Internet access over the phone line. Some customers may have more advanced PC and broadband capabilities like cable or ADSL
Customer base	Up to 100K subscribers in the first year, growing to 500K over 2-3 years period, up to 700K in 5 years
Database	New Operational Data Store or Data Warehouse needs to be set up. Volume of primary user data (without indices and copies) estimated at 4TB
Security	Only authenticated and authorised customers have access to the service. Authentication needs to be stronger then basic password protection (eg. Digital Certificates)
Privacy	Customers are allowed to see own records only
Subscriptions	Service will be provided on subscription basis. Application allows various subscriptions with different subscription fees
User Interface	User Interface adheres to the Enterprise and industry guidelines for look-and-feel, useability and accessibility. Application requires from customers only basic computer and Internet navigation skills
Availability and Failover	Application shall be available for customer access on 24x7 basis, with the ability to bring application back online in case of failure within 1 hour. Overall availability should not fall below 99.5% a year
Concurrency	Application should expect 400-500 concurrent user sessions routinely. Users may request on average 4-5 statistical or mapping queries per session against their records
Online	Application should ensure acceptable latency for queries when customer accesses services over the Internet, even without the enhanced bandwidth of cable or ADSL
Statistical Reports	Application will query Data Warehouse (or the Operational Data Store) and render the response on the customer's browser as tables, graphs, bar charts, pie charts and other graphical views
Geographical Mapping Reports	Some customer queries will generate geographical maps with customer's statistical data shown as map overlays. Customer will zoom and pan geographical maps for better business analysis
Data Sources	Application receives periodical data feeds from several Enterprise back-end systems. Also, application will require access to personal and business directories, demographical statistical data and categories, and geographical map layouts
Platform	Enterprise is open to the platform selection for the solution. Enterprise has built its online presence on the Windows and Unix platforms. However, existing presentation and business logic layers in the Enterprise lean towards J2EE-based solutions
Budget	Business earmarked certain dollar amount to this application, based on budgets allocated to somewhat similar online applications. Feasibility Assessment shall include rough blueprint for the solution with better budget estimates

High-Level Requirement	*Qualified/Quantified Requirement*
Timeframe	Business requires deployment of the application in 5 months. Business plans to unfold the marketing campaign to raise customer awareness and to meet the aggressive customer take-up schedules, in anticipation of the coming revenue from the application

At this point, Enterprise Architect should be able to find the best combination of technologies and products that will implement the cohesive solution end-to-end – by looking at what is required and what the Enterprise has got.

Enter the maze of practical compromises and challenges in the Enterprise Architectures, where:

- There is no black and white, absolute right and absolute wrong
- Technology has its limits
- The 'best' or more expensive technology does not guarantee that you've got the solution. 'Fit for purpose' is the flavour of the month
- Technical decisions are only half of the architecture solution at best. Rest of it – building relationships with all stakeholders in the solution, communicating and actively promoting technical decisions, leadership, alignment with other enterprise components and processes
- Solution does not exist in isolation – 'big picture' has the same impact on the solution (if not greater) than the 'small picture'. And the 'small picture' in the enterprise is not that small either
- Everything seems to depend on anything else
- Nothing is perfect in the long run or on the big picture
- Nobody is happy with everything. Someone bound to be very unhappy with something
- Everybody may have own opinions, preferences and agendas
- Right people in wrong places at the wrong time, and any other combination of *wrongs* and *rights* above. Wrong people with best intentions and wrong expertise may be very enthusiastic, vocal and proactive to the detriment of the project. Right people with right ideas may be passive or silenced in the chorus (which makes them in the end the wrong people for the architecture task - think about it!)
- The best technical solution does not necessarily gain a sufficient political clout to be accepted and funded – technical guru may not have muscle or enough support in the enterprise, and decision makers may not be so sure
- Expected benefits are measured against the risks and costs…

This rather disparate list of compromises goes on and on, but you've got the picture.
Scene is set where you get your room for manoeuvre or elbow room, the environment that you operate in…

Enterprise Architecture is not just a bag of the 'best' components, but the right components in the right balance.
Complex technical solution is not solely the kingdom of the technical expert anymore. For the solution to work, it has to be sold to and bought by all stakeholders in the build, implementation and use of the solution. Enterprise Architecture takes its shape in the collaborative process.

Fusion of Software Engineering methodologies with the Enterprise culture, infrastructure and practices has an immense impact on how (and with what difficulties and frustrations) practical compromises will be found. For the Enterprise Architect, this is all part of the fun – comes with the territory.

Having paid dues to the organisational aspects in the 'big picture' of the Enterprise Architecture (they will be covered in the section on methodologies), let's abstract from this part of the story for now and concentrate on the pure clean fun – technical architecture.

Figure 6.1 shows 3-tier high-level architecture for the solution.
At this stage, we are trying in our drawings intentionally to get by with the rudimentary intuitive notation that did not achieve an international recognition yet (to say the least), but explains our train of thought just fine.
We explain the more appropriate and universally accepted UML notation in a non-threatening pragmatic fashion later in a book.

Actually, Figure 6.1 mixes both logical and physical high-level views in one diagram. We demonstrate:

- Main logical layers in the 3-tier architecture. We shall see in the following sections that solution, as it unfolds, will have more than three tiers
- Mapping of the logical view onto physical deployment configuration - we start small by deploying the whole solution onto the single box. (We assume here that all, yet to be chosen technologies and products for the application end-to-end, allow such a deployment. I.e., all components can run on the same type of hardware and operating system in the first place)

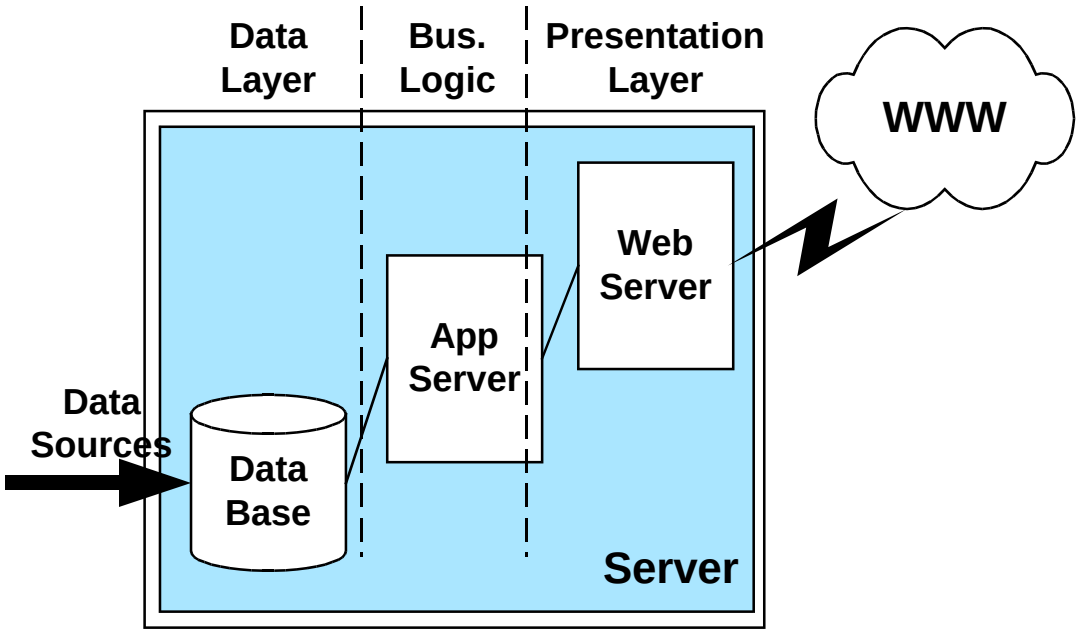

Figure 6.1. *Architectura Sapiens* Evolution, Take 1

Solution calls for the following major components in the architecture:

- ***Web Server*** – Internet-facing front-end of the application. Customers will gain access to the application by pointing the URL on their browser to this Web Server, and initiating the HTTP/HTML request/response communication dialog with the application. Presentation layer is represented by Web Server and some components of the Application Server responsible for the rendering of HTML pages and maintaining the HTTP session
- ***Application Server*** – layer or component handling the business logic of the application and supporting the concurrent access to the application by many customers simultaneously. At this stage, we throw all business logic functions into this single high-level component. However, we can see already that in addition to security, subscription and query housekeeping logic, we have two very specialised sub-systems that warrant special attention:
  - ***Report Formatting*** to produce graphs of statistical data
  - ***Geographical Mapping*** to provide mapping reports functionality
- ***Data Warehouse or Operational Data Store*** – data layer of the application responsible harvesting, validation and timely load of data for the online reporting

These mayor application components determine streams of work on the project, as well as decision points for the technologies and products.

We will have some decisions to make on selecting the most appropriate product for every component (best amongst the comparable competitors, and better suited to the architecture end-to-end in the context of our particular enterprise).
We will glide through this process by pointing to some common scenarios of providing the due diligence and justification for the technical decisions.
Your justification for recommended products may range from the compliance to the enterprise and industry standards and guidelines, to de-facto enterprise infrastructure realities, to formal evaluation of competing products head to head for the best satisfaction of the application requirements and multitude of the quality criteria.

Fortunately, IT market provides us with technologies and products of all shapes and sizes. We should be able to pick off-the-shelf component that does most of the job for any part of our application. We do not have to invent a wheel.
Selections in our fictitious example have been made purely for demonstration purposes only. Real mature products have been mentioned here, and any of them can do the job in any combination, with more or less development or integration efforts. Specific decisions will be influenced by the context of your enterprise and requirements of the application.

HTTP Web Server is a standard piece of Internet-facing architectures. Mayor products include Apache, iPlanet and Microsoft IIS.
Most of Web Servers can run on both Wintel (Windows+Intel) and variety of Unix platforms, excluding MS IIS, of course.
Our enterprise has an existing deployment and expertise with iPlanet Web Server running on SUN Solaris platform. Furthermore, some security and access control features have been implemented, including Digital Certificates and coarse-grained authorisation. Application should be able to leverage this expertise, and even existing deployment if capacity permits.
So, iPlanet Web Server on SUN Solaris it is.

On the Application Server front, Enterprise decided to standardise on the J2EE-compliant WebSphere Application Server from IBM across the whole online space. It makes a lot of sense to streamline technologies, expertise, resources and deployment. Large enterprise can achieve big savings and good level of re-use and agility by doing this.
There are numerous other J2EE Application Servers that can ‘run anywhere’, as well as strong offerings from Microsoft. However, enterprise guidelines made the decision easier for us.
So, WebSphere J2EE Application Server on SUN Solaris it is.

Web report formatting component has several possible good solutions, both off-the-shelf products and in-house development.
Some possible solutions include Crystal Enterprise and Crystal Reports from Crystal Decisions, Alphablox from Alphablox, JReport from Jinfonet, jClass Java class library from Sitraka or in-house Java development, possibly using Java applets and Java 2D GUI package.
Our application development team embarked on evaluation of products from Crystal and Alphablox. Team performed comparative analysis of both products head to head against general quality criteria and specific application requirements. Both offerings could do the job – no big gaps or showstoppers were discovered. However, some previous decisions on J2EE WebSphere Application Server tipped preferences towards Java-based Alphablox in the context of our application. Team expects to get additional benefits from cohesive architecture end-to-end and better alignment of component technologies.
So, Alphablox on SUN Solaris it is.

Geographical Mapping solutions on the market are well represented by such products like ArcIMS from ESRI and MapInfo from MapInfo. Both products could do the job for our application.
One of the stakeholders in our project came with clear preference towards ESRI ArcIMS.
Coincidentally, this is Java-based solution as well – we are in luck.
So, ArcIMS on SUN Solaris it is.

Data layer in our application requires use of the relational DBMS.
Several major contenders come to mind – DB2 from IBM, Oracle from Oracle and SQL Server from Microsoft. Our enterprise has got a corporate, all-you-can-eat, corporate license for Oracle and the extensive expertise. Again, life made easy for us.

Still, decision to go for Oracle contained to the application online database. Data Warehouse back-end may require different tools for the ETL processing – we postpone this discussion till later.
So, Oracle on SUN Solaris it is.

Now we have selected technologies and products for building our application. Note how well aligned our chosen products are – they all can run on the same platform, and even on the same box if we want to. Figure 6.1 demonstrates exactly that.
Such cohesion is not a small achievement by all means. We can scale and grow from here in many ways, and that will be demonstrated in the following iterations of the solution design. Conversely, we can implode the whole deployment onto the single box, and achieve good manageability and savings in the development and unit testing. But let's take one step at a time, shall we?

We need to review our architecture design against the business requirements.

What is wrong with this picture above? Nothing, it will work. However:

- Application is as available, as the single SUN Solaris box – we've got one big single point of failure without the hot stand-by or ready replacement
- Existing secure Web Servers are running on the separate boxes at the moment, somewhere in the corporate firewall. Security guidelines will not allow deployment of the application in the DMZ or outside the firewall. Application will have to deploy its own secure Web Server, or proxy Web Server, collocated with the rest of the application components. This won't be easy
- If our server is hacked, we expose all components including customer's private data to the intruder
- Single SUN server will have to handle significant workload. We have to aim for the high-end machine
- Scalability is limited by the power of the single server – we will have to rely only on vertical scalability by brute force
- Problems and performance degradation in any component will have a direct impact on the whole application. Any component may choke the whole application
- Upgrades or maintenance of hardware or any of software components will likely cause downtime of the whole application

We are not quite there yet in satisfying the requirements of our application.

At this stage, we should have a reasonably good idea on feasibility and viability of the solution, including costs. We assume that business model for the solution is perceived by business as feasible, and business gives the team go-ahead to proceed with the detailed design and build of the solution.

Solution is taking shape

Figure 6.2 shows the next incremental improvement of the application infrastructure.

We took notice of the comments from our security folks – application will connect to the existing Web Server in the firewall, and the rest of the application will be deployed on a corporate network behind firewall.

Also, we realised that our application is very IO-intensive. We moved Data Layer onto separate box.

Furthermore, we found that Data Warehouse back-end performs extensive ETL processing, in its own time. This may clash with availability of the online database.
Online database needs to be optimised for heavy workload from online request and be available as the application itself – '24x7'. Online database is mostly read-only.
Strategic data management decision has been made to decouple the Data Warehouse and the Online Database, and to deploy them on separate boxes. Online Database will be refreshed with updates from the Data Warehouse through the publishing process with negligible database downtime at the acceptable pre-defined time, when online usage is minimal.

Whole application deployment is partitioned across four boxes now.

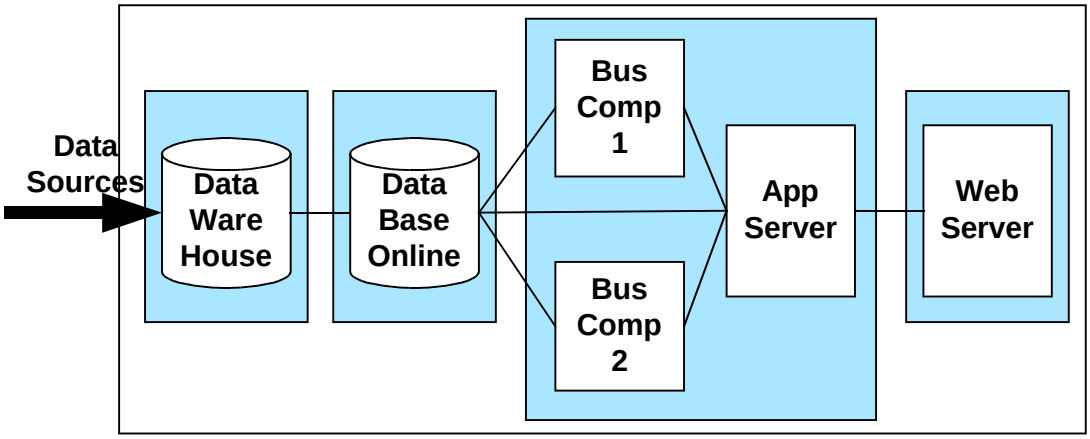

Figure 6.2. *Architectura Sapiens* Evolution, Take 2

We have achieved better partitioning and resource management in our application, and satisfied security requirements.

What is wrong with this picture above? Nothing, it will work. However:

- Three main components on the business logic layer are still on the same single box. They compete for the same resources. Any upgrade or maintenance brings down the whole box, and the whole application with it
- Any of the four boxes provide a single point of failure for the application
- Failure of any box may take at best few hours to recover and to bring the application back online – this is even in case of spare server available on cold stand-by
- Data storage for Data Warehouse and Online Database should provide large capacity, reliability and flexibility of data management procedures. Our application warrants more advanced storage solutions than plain hard disks. We have to consider RAID, NAS or SAN

Again, we are not quite there yet in satisfying the requirements of our application.

Getting there…

Figure 6.3 shows next improvement in the architecture design for our application.

Report Formatting and Geographical Mapping solutions are themselves quite resource-hungry. We decided to deploy them on dedicated boxes (Business Components 1 and 2).
Capacity planning may be performed in a more flexible way by better satisfying the need for resources of every separate component – Application Server, Formatting Server, Mapping Server.

Furthermore, availability has improved somewhat. If Formatting Server is down – application as a whole is still online, only graph reports are temporarily unavailable.
The same goes with upgrades and maintenance. If we are prepared to sacrifice some functionality temporarily, we can administer Formatting and Mapping Servers judiciously while the whole application is online.
Apart from this questionable benefit, every server is a single point of failure still.

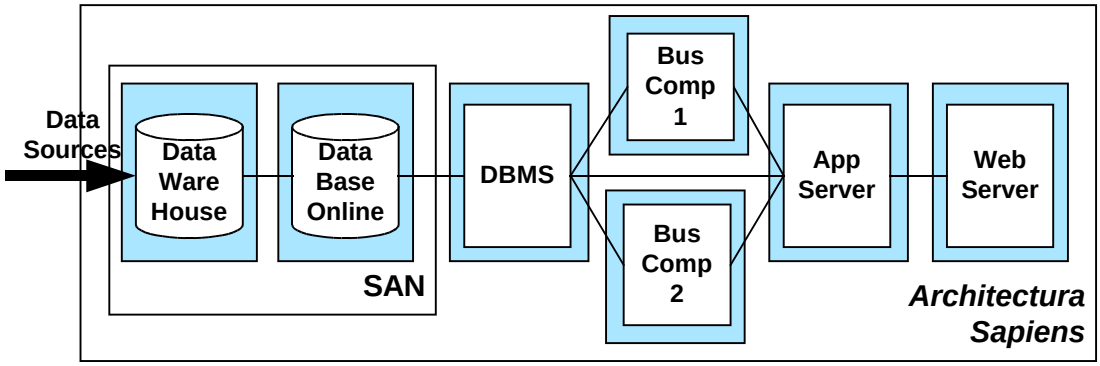

Figure 6.3. *Architectura Sapiens* Evolution, Take 3

We achieved better resource management and availability through further partitioning. Application leverages SAN storage that is shared by many enterprise applications.

What is wrong with this picture above? Nothing, it will work. However:

- Availability and failover still do not satisfy requirements of the application
- Scalability options are still limited

Enterprise-grade solution

Figure 6.4 shows further improvement of the architecture.

High availability and horizontal scalability requires some kind of replication, redundancy and load balancing. In general, Load Balancing may be achieved in two ways – software and network hardware.

Luckily, every component of our architecture has got Load Balancing capabilities. We intend to eliminate every single point of failure one by one.

Several instances of Web Server may have the same URL and be configured on network for Load Balancing. Routers spray HTTP requests to the next available Web Server.
Router will maintain the 'sticky' session by ensuring that the next HTTP request from the same HTTP session is directed to the same instance of Web Server.

IBM WebSphere has got software Load Balancing solution of its own. We use WebSphere 'cloning' mechanism. Cloning is done by configuring the WebSphere plug-in on Web Server to point to several instances of WebSphere, and thus plug-in will spray HTTP requests to the next available instance of WebSphere.

Alphablox and ArcIMS both have got software Load Balancing solutions based on Java servlets.

Every online component now can have redundancy. We can replicate and scale every component independently from any other.
If one instance of WebSphere server goes down, other instances take over workload. The same goes for the Web Server, Formatting and Mapping components. Application stays online without the visible impact on the customer.
If workload on the Mapping component unexpectedly increases, we deploy additional Mapping Server without bringing the application or any other server down.

If we need to perform maintenance on any online server, we disconnect it and return back into service when we done, without an impact on the online processing.

We achieved high availability, manageability and full horizontal scalability – for each and every component separately.
And we still can scale vertically as well – by applying the brute force and upgrading any online server in our application.

We are in heaven now. Skies are the limit. And, as you are still with us, you must have reached *nerd*vana[1].

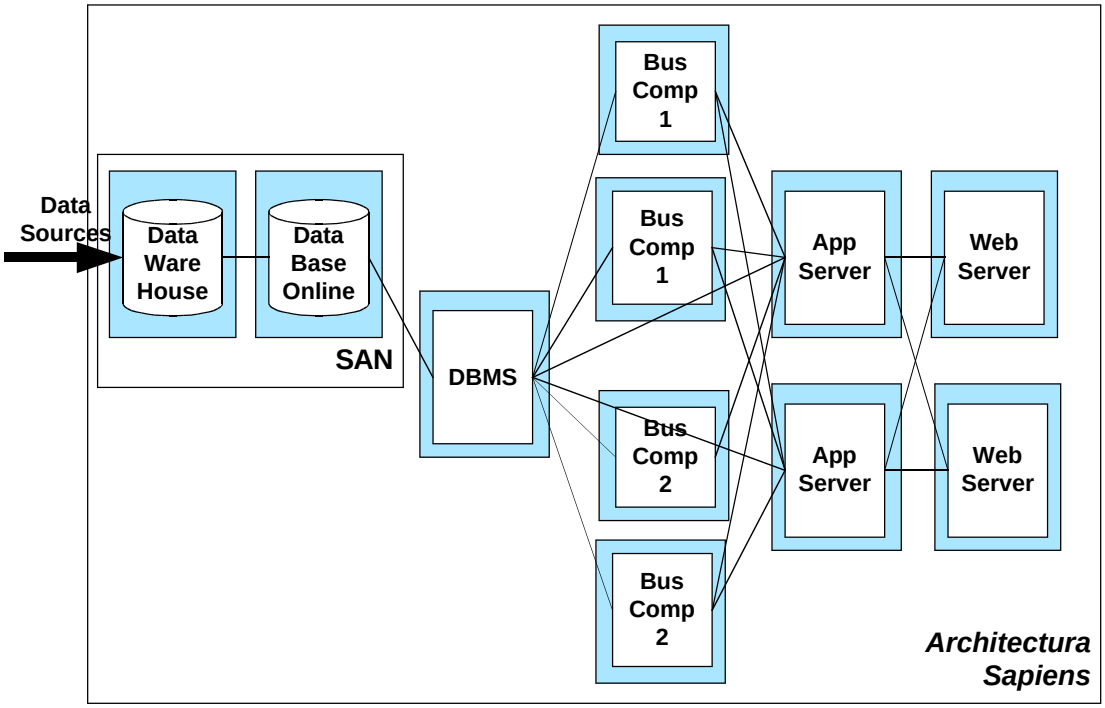

Figure 6.4. *Architectura Sapiens* Evolution, Take 4

Figure 6.5 shows our design increments side-by-side, from the initial rudimentary deployment blueprint to scalable and robust industrial-strengths solution.
Revisit this case from time to time when you read the chapters on protocols and technologies in the Architect's Toolbox.

[1] Word invented by Scott Adams, the creator of Dilbert [WWW, Dilbert]

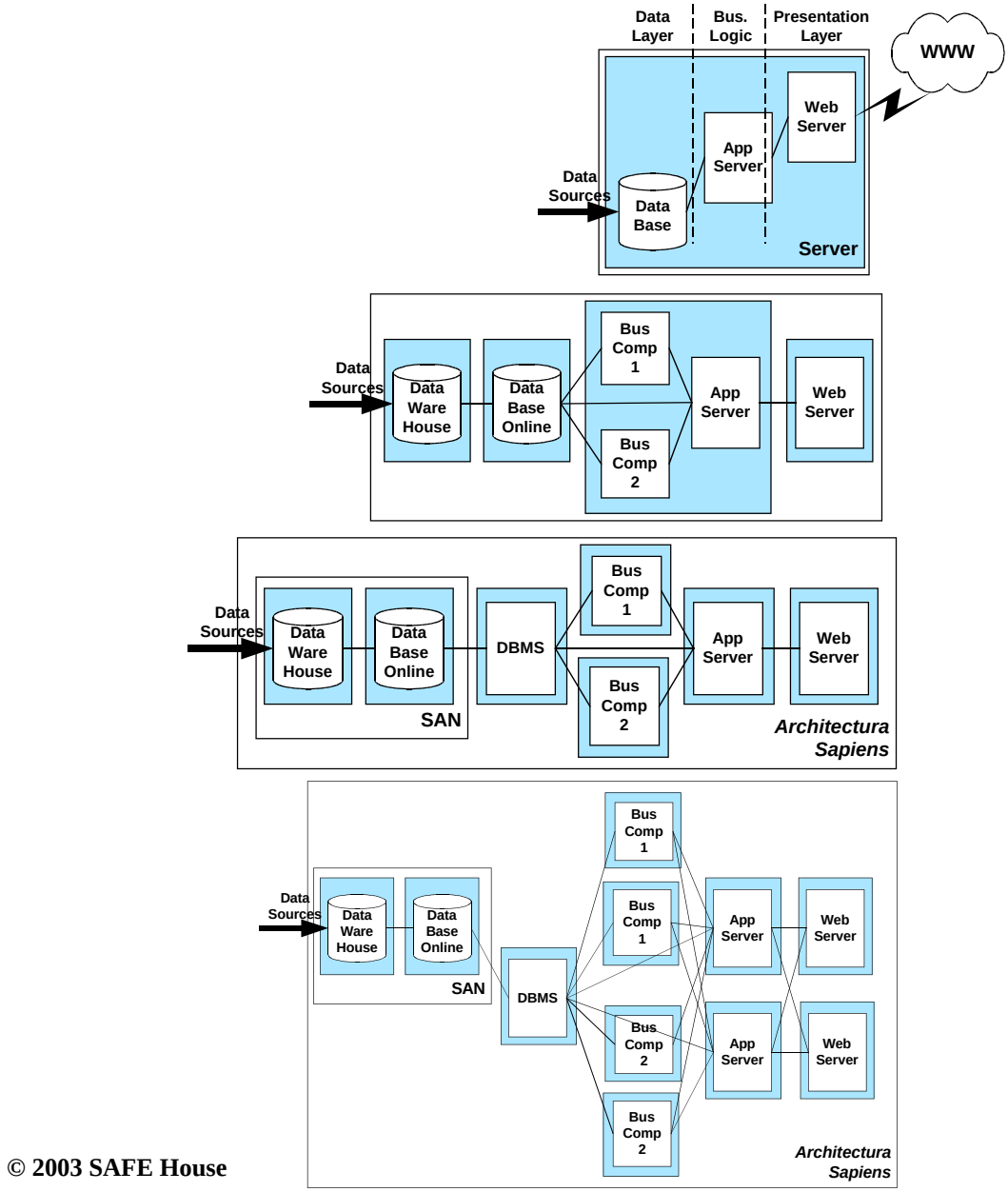

Figure 6.5. *Architectura Sapiens* Evolution, Stages Side-by-Side

<<< ... >>>