The Principles of Teleinformatics

Duncan Stonebridge
duncan@softengines.co.uk
013398 81139/07740 642457

M Phil Thesis

December 2004

Abstract

- The Network Effects described by Metcalfe's law explain the exponential growth experience by the Web and the far reaching economic impact and current climate of organizational change the implications of digitization, mean that many transaction cost are now lower and that there will more outsourcing and disintermediation.

- Knowledge Management assumes all systems and data are used in integrated and collaborative way. Sharing Knowledge involves dynamic and heterogeneous distributed computing systems. The challenge in shared Knowledge discovery is achieve trust and consensus in the collective view of the data and information constituting the knowledge. The purpose of knowledge management is to reduce overall documentation through identifying redundant content and streamlining its production, there is now the potential to move towards more semantic mechanisms for controlling and managing information. There is a clear need for an intuitive user interface to data structures capable of reflecting complex situations.

- The systems managing that organizational Knowledge will increasingly need to be decentralized and collaborative as well as scalable and extensible. With these requirement those systems become more complex and burdensome to manage and administer. Discovering and conceptualised the structure and properties of information systems is intrinsically difficult, particular for large systems that are subject to arbitrary change. The aim is to develop the tools and techniques to enshrine extensibility into the knowledge management systems, and to allow bottom-up design and development to occur in a manageable way. The choices of tools and techniques required for the construction and developed of systems for knowledge sharing will be determined by the requirement to interoperate across heterogeneous systems, and therefore have the qualities of openness. Open distributed systems such as the Internet are necessarily scalable and extensile

- RDF provides a framework for describing and interchanging Metadata. Using XML it can be used to describe resources without a priori of the semantics. The capacity to develop machine interpretable vocabularies is the essence of the Semantic Web. These metadata applications can then be accessed and used by intelligence agents in knowledge discovery, extending the current World Wide Web beyond a publishing metaphor into global knowledge base.

- We defined define the processing and management of distributed data, metadata, information and knowledge representation as *teleinformatics*.

Contents

Acknowledgments

Firstly I must acknowledge the input of my supervisors Dr. Heath James and Prof. Ken Hawick and thank then for guidance, inspiration and encouragement. Also I would like to thank my Member of Parliament, Sir Robert Smith Bart. for his assistance and support with certain bureaucratic matters. Finally I must thank my wife Anni for making this work possible, whilst undertaking this entirely self-funded project, on top of the usually moral support she was the household's only breadwinner, and should be gratefully acknowledged for her sacrifice and patience.

Authors Declaration

This work contains no material which has been accepted for the award of any other degree or diploma in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. I give consent to this copy of my thesis, when deposited in the University Library, being available for loan and photocopying.

1 Perestroika

1.1 Introduction

Recent years have seen a proliferation of network available data. The growth, in that data has been in both volume and complexity and matched by a growth in usage. The Internet and World Wide Web have undoubtedly been successful. At the turn of the millennium, the potential of e-Commence, with and a promise of revenue generation was heavily invested in. Disappointment with the profits of the Dotcom boom contributed to the current downturn in the IT industry. The focus, in business, industry and commence now has shifted on to maximising existing assets and cutting costs, with the emphasis on streamlining processes and better integration and interpolation of existing data resources, from which usable knowledge can be extracted and managed. Data Mining [67, 68] is the term given to this process of knowledge discovery. Pivotal to the subsequent Knowledge Management are the processes of collaboration and communication.

Sharing Knowledge within e-Working communities assumes a dynamic and heterogeneous distributed computing system. The challenge is to create consensus between different implementations of a universal conceptual data model. Internal representation of a conceptual data, such as those in a Relational Database or other Information management system, will be engineered at different times and by different organisations, collaboration between different developers is not always practical. A solution is for open, platform agnostic and vendor neutral, standards to facilitate Information interchange between independently developed and managed systems. This approach is typified by XML (extensible mark-up language). Web services, which are built upon XML and open standards, makes use of tried and tested protocols and mechanisms for electronic data exchange, storage and retrieval within widely distributed computing environments that can be potentially secure, reliable and inexpensive.

1.2 Aims

This work has been largely motivated by the authors own commercial experience in IT systems support and development, working predominately at what could be considered as the back-end of the systems development life-cycle. That is during the use and review stage of a system. This involves the implementation and evaluation of new software solutions and the integration with any existing legacy systems and data.

The introduction of a new system inevitably involves the discovery of errors and omissions and the consequent debugging or changes to code and data structures. Over time, as the system is used more and matures these bugs are gradually, in theory, eliminated. In practice this process can extend for into the life time of the system. Through usage, areas of improvement and redesign can become evident and together with new and different requirements the whole systems development life cycle can begin again. The reality as described in much of the literature is one where software systems are continually modified and added too. Along with interfaces and functionality, data schema can also require extending.

Another issue for many Information Systems is that of integration. The information stored within a database can represent time, effort and an asset to the organisation who will therefore be keen to preserve its legacy. Integration of different systems can also reduce replication and redundancy and increase the usefulness of the overall integrated system.

Increasingly in the current changing economic climate there is a need for inter-enterprise systems, as will be evidenced. Also multi-agency partnerships are a common feature in current UK political thinking. There is much literature that speaks of the need for different organisations to exchange data and information. The technical challenges that arise in this B2B (Business to Business) information interchange are those of achieving trust and consensus, these are on the one hand, security and reliability considerations but also issues of integration and interoperation between differently designed and implemented systems. E-Business implies that there is no overall central control and the information interchange must cross administrative boundaries as well as counties and continents. What is also implied is that is there is a common domain of knowledge shared by the e-

Business coalition, a common conceptual data model of which they possess different internal representations.

There have been two areas of frustration working at the coal-face of data and information management. Firstly there is the problem of modifying and extending systems, and secondly there are issues of concurrency between different or distributed datasets, where discrepancies can be all too obvious. These problems are exacerbated by both complied hidden code and remote access difficulties. For a Database Administrator the challenge is to understand, modify and extend schema, the anecdotal experience is that these grow in a disorganised fashion which is often poorly documented, and sometimes not at all. The difficulty arises from the fact that the underlying relationships and structure are generally not machine readable and will often require human understanding. Schema understanding and discovery is also a prerequisite of database integration. If this is the case then the further problem of concurrency and redundancy needs to be addressed. The constituent sub databases of a multidatabase share the same conceptual data model, they have different internal representations of either different parts of the data model or they overlap, here transactions on same parts will occur at the different times or not be replicated. The necessary coordination can not always be centralised and can prove a burdensome administrative challenge. For distributed and decentralised information systems whether they be Relational Databases or semi-structured and unstructured documents, the desire is for a single moment of truth to exist, that is there should be a version of data that is correct and up to date.

It is the goals of Web Services, built upon XML (eXtensible Mark-up Language), to achieve extensible schema that can be interpreted by both man and machine and shared openly in distributed systems. This work is an attempt to investigate the promise of web services and arrive at a framework for building collaborative semantic webs for sharing knowledge in today's increasingly inter-enterprise and dispersed multi-agency partnerships.

This work aims to critically investigate and evaluate the tools and techniques for building collaborative and extensible data management systems, reviewing the literature to analyse and to identify the relevant methodologies

and identify any new practical considerations, technology choices, design issues and requirements for those attempting to realise the overall concept.

1.3 Knowledge Management

Knowledge Management is defined as the art and science of collecting organisational data, and by recognising and understanding relationships and patterns turning it in to usable, accessible information and valuable knowledge [57]. Data is an item or event out of context with no relation to other things. Information is relationships between data and possibly other information. Knowledge is represented by patterns among data, information and possibly other knowledge. These patterns do not actually constitute knowledge until they are understood. Wisdom is the recognition that knowledge patterns arise from fundamental principles and the understanding of what those principles are.

The Knowledge Management paradigm represents a holistic approach to an organisations data and; information assets, a convergence between business practice and IT management. It is a perspective for implementing organisational change, which gets people to record knowledge (as opposed to data) and then share it.

The implications of the adoption of a Knowledge Management strategy for an organisation is that the processing of data and information can be combined with a qualitative understanding, and then be shared and communicated. Such a strategy need not depend on any overarching information processing system, whether new or existing. Content Management is an integral aspect of such a strategy. This is the integration of structured and unstructured informational assets through out the organisation, and then making then accessible to the relevant Knowledge Workers. These are defined as those undertaking mental work to generate useful information. [71]. Knowledge workers access data use knowledge, employing mental models as well as requiring concentration and attention. Examples of these workers include, professors, writers, financial analysts, systems analysts, managers, accountants and lawyers etc. These, usually professional persons, bring creativity, problem solving skills and understanding, through having expertise, in their particular domain, of processes and procedures. Productivity for these worker can be improved through improved motivation,

task management and reduction of errors, omissions and redundancy, as well as conserving attention and concentration, all of which can be contributed to by effective Knowledge and Content Management.

For the organisation or enterprise as a whole, effective Knowledge Management can contribute success through:

- Identification of waste and inefficiency, streamlining processes and improving productivity.
- Identifying and targeting of customer preferences, improving service and gaining new custom to consolidate increase market share. (Systems for this purpose are known as Client Relations Management).
- Identifying best practice, and sharing that throughout the organisation. Learning Management Systems can be responsible this aspect.
- Decision support systems, informing strategy and helping in management decisions.

One conclusion that can be drawn is that an organisation, enterprise or any other community of knowledge stakeholders may possess collections of systems for managing their data and Information, such as Client Relationships Management, Learning Management and Decision support applications. They however, do not have Knowledge Management until all systems and data are used in an integrated and collaborative way, where domain expertise can be utilised to derive understanding. To this end Data Management and Information Technology Management is important.

1.3.1 Challenges of Knowledge Management

Challenges of knowledge management are not merely technical but also involve the policies and attitudes of the organizational structures that such a system must reflect. The general problems with the underlying data management systems are discussed in [102, 70]. These problems typically include poor control and management of data, difficulties in interfaces and interacting with data, and the problems of accessing distributed data storage, such as latency issues and integration problems. Redundancy is

another significant problem with data management systems, such systems according to [102] have grown haphazardly with essentially the same data being found within different stores, with obvious dangers of amendments not being carrying over to all the systems involved. Clearly these are both technological as well as administrative issues which can be alleviated through good modelling and management of data. Another challenge mentioned by [102], is the potential for a lack of reality due to the complexity of knowledge that the data is trying to represent. Also the expectations of end users in interfacing the data are discussed, including the desire for quick results in a form that can be understood and interface dialogues that can remembered, even by an occasional user. There is a clear need for an intuitive user interface to data structures capable of reflecting complex situations, from which users will desire a timely performance.

In [79] issues of personal productivity are addressed. Where there is a paradoxical situation where the potential benefits of IT investment can be lost in an organisational setting. The issues and possible explanations include the learning curves involved with new interfaces to new systems and iterations of versions, also leading to compatibility issues and also the addition of increasing numbers of largely unused features. The proliferation of information within organisations is also discussed by [79]. Much of this information is largely unproductive, produced in an organisational climate that demands individuals make a good impression even with internal documents. The Word Processing and Desk Top Publishing applications available can ironically impact on productivity, through excessive revisions, rewriting and the inclusion of unnecessary presentational touches. Here the suggestion is of Information production rather than use is the problem, the challenge of knowledge management is therefore to reduce overall documentation through identifying redundant content and streamlining its production wherever possible. Much of this is clearly an issue for management and involves challenging organisational culture. However there is now the potential to move on from merely creating and publishing information and content towards more semantic mechanisms for controlling and managing that information, where style can be more easily separated from content with benefits in accessibility.

1.4 The Web

An example of a content and knowledge management system was developed at CREN, the European Centre for Nuclear Research. In 1989, the physicists required a means of sharing documents. By using documents created in hypertext and accessible over the Internet, the World Wide Web was invented. They created the Hypertext Transfer Protocol (HTTP) for downloading the documents from specific locations, identified by Web Addresses, the (URL) Uniform Resource Locators. Hypertext Mark-up Language (HTML) was language created to format the data and information in those documents, which possibly include links to other related documents, thus making a Web of interlinked information. The concept of hypertext can be traced back to 1945, and Vannevar Bush and his article "*As We May Think*" which descried a "machine for browsing and making notes in an extensive online text and graphic system". The term *Hypertext* was coined in 1974 by Ted Nelson, along with the concept of *Hypermedia*. Academic institutions experimented with hypertext and it was not until 1987 that the concept was introduced to public, when Apple marketed Hypercard, and both Apple and Microsoft introduced help file systems, including clickable links to aid navigation. In 1992 CERN introduce the Web to the Internet community [18].

The Web's success is attributable to its *open system* nature, in which it can be extended and implemented in new ways without disturbing existing functionality. The responsibility is on independent vendors to produce browsers compatible with the agreed and published standards. Web browsers such as Netscape's Navigator and Microsoft's Internet Explorer are widely and easily available, as are the tools for creating Web pages. These browsers read the Web pages and display the text and graphics as dictated by the mark-up tag usually invisible to the user. It is this extensibility and scalability combined with the globally accessible and interconnected nature of the Internet, which has lead to the webs exponential growth within the last decade.

1.5 Computer Networks

1.5.1 Advantages of being Networked

Metcalfe's Law, attributed to 3Com founder and inventor of the Ethernet Protocol, Bob Metcalfe, states, according to [99] that "The power of the Network increases exponentially by the number of computers connected to it". The power, described by [98] as "The usefulness or utility", equals the square of the number of users. The logic being behind this law is that for every computer added to the network both uses the network as resources as well as itself being a resource. Reed's Law follows on from Metcalfe's [97] in describing the utility of large networks, particularly from a social perspective, which will scale exponentially with the size of the network. With Reed's law the number of possible Sub-groups of network participants is 2^N Where N is the number of participants. This figure will grow more rapidly than N of the number of possible pair connections N (N+1)/2.

A description can be found in [97] of the Network Effect phenomena. This is a more general observation of the economic positive externality of being part of a network, an externality being an indirect effect. A Network Effect is where the value of a product, any particular goods or service, is added to by new owners of that product. Similar to Metcalfe's Law the overall value of a particular product possessing a network effect is roughly proportional to the square of the number of users or customers. The telephone is prime example of a product with a network effect. According to [97] purchasing a Telephone make other peoples Telephones more useful. Metcalfe's law and Network Effects imply that a technology needs to achieve to a critical mass in order to become truly successful, according to [99] at the critical mass of users achieved for a technology the wider social, political and economic systems change as had happened with the Internet. In the case of the Telephone, there was a prediction that dictatorships could not survive beyond a particular critical mass of telephone ownership. This was borne out by the case of the Soviet Union, which collapsed around the time this critical mass was reached. These effects have underpinned many of the dotcom strategies [97] of increasing market share even without regard for profitability, such as Freeserve's emergence on to the market as a free ISP, eventually charging for its services after gaining a market share. The Network Effect does have a ceiling, when congestion occurs, in a phone system for example, where the

technology fails to scale with growth, the addition of additional telephone users could represent a degradation of service.

The Network Effects described by Metcalfe's law in conjunction with Moore's Law help explain the exponential growth experience by the Web and the Internet and their far reaching economic impact and current climate of organizational change.

1.5.2 Computer Networks

Distributed Computing covers a wide area encompassing disciplines in electrical engineering and computer science, including computer networking and internetworking, metacomputing, knowledge sharing and multidatabase. The use of multiple computers whether separated by a few feet or by continents present many advantages along with challenges. These advantages include multiplying processors, increasing data storage, making back-up data copies and of cause information interchange and knowledge sharing between users. Networked computer users can share computing resources such as printers as data storage with the use of file servers. Not so long ago many users shared processors via networked dumb terminals.

Computer Networks are fully discussed in [43]. These networks are built from interconnected hardware devices (routers, switches, bridges, hubs, repeaters etc.). Theses are linked range of with materials capable of the transmission of information such as wires and cable for electrical transmission, optical fibres and wireless systems using radio waves. Computer networks also consist of software, in the form of protocol stacks, communication applications and device drivers.

A computer network can be regarded as a communication subsystem, defined in [43] as a collection of hardware and software providing the communication facilities for a distributed system. This subsystem is comprised of nodes and hosts, subnets.

- A *node* is any computer or switching devices attached to the network.
- A *host* is defined as any device, such as a computer, that uses the network for communication purposes.

- A *subnet* is a unit for delivering data or routing, it is a collection of nodes that can all be reached on the same physical network.

Computer networks are often classified in accordance with size and purpose, including:

- LANs, local area networks, such as those operating within a single location such an office building or university campus.
- WANs, Wide area networks, linking different buildings or campuses, crossing a city or further.
- MANs, Metropolitan Area Networks these are wide area networks for specific purposes such as cable TV.
- Wireless networks, use radio transmission and can be for spanning different devices that could be in the same room, or much further distances across cities and countries.
- Internetworks, connect different networks effectively by making one bigger one.
- VLANs, Virtual Local Area Networks, create the appearance of a local area network out of LANs and WANs internetworked.

The Internet is itself a single communication subsystem providing data exchange between all the hosts that are connected to it, comprising of many subnets, its infrastructure specifically designed to integrate those heterogeneous subnets.

The Internet and the computer networks commonly in use today have their origins in the cold war of the 1960s. The command, control and communication structure of the nuclear era needed to hold out against repeated and devastating strikes. An architecture that crashed because of a first strike on the Pentagon, or a commando with a pair of wire cutters, obviously would not allow retaliation and render the nuclear deterrent ineffective. A partially functioning system would, therefore a distributed and decentralised approach was taken, one tolerant to degradation of service. If a communication centre or a link was destroyed then the surviving ones communicate via alternative routes. The next step went beyond the capabilities of the old telephone and telegraph circuit switching networks. Packet switching some-times referred to as a *store-and-forward network* exploits the data storage capacities of computers. Packet switching can be

regarded as virtual postal systems where the packets are sorted and distributed to the appropriate destination. The overall message is divided into packets, this consists of a fragment of data and address of the destination as well as the sending host machine. Countless millions of theses digital packets flow through the internet, being cached at nodes and routed from host to host. The use of a check-sum indicates the correct transmission of any message lost or corrupted messages are resent. Restricting the overall packet size allows for sufficient buffer storage at a recipient machine and prevents channels being blocked by excessively large packets. The result is a network which arbitrarily grows, changes and continues to function in the presence of individual technological failures.

The challenges in building a distributed computing architecture include heterogeneity, openness, security, scalability, failure handling, concurrency and transparency. Any such systems provide a certain level of performance and quality of service as well as dependability and fault tolerance. The various models of distributed systems are discussed in [43]. These include interaction, security and failure models, client–server models and variations such as multiple servers, proxy servers, and peer processes.

The Internet is a well known widely implemented distributed computing system. Its basis is TCP/IP, Transmission Control Protocol / Internet protocol, as fully discussed in [43]. This solution to internetworking provides the foundation for the Web (HTTP, Hyper text transfer protocol), Email and FTP (file transfer protocol). Transparency of issues such as access, location, performance, concurrence, replication etc, hide the technical minutiae, allowing the developer of applications to concentrate on logic and usability. Security has long been seen as a vulnerability of such open and remotely accessible systems, encryption enables private channels to tunnel through the internet building virtual private networks (VPNs). Secure socket layer SSL is a tried and tested approach designed to work with HTTP, this is widely seen in eCommerce. A developer of any web application can these days afford a certain transparency of security.

The mainstay of what we regard as the web, HTTP, is an example of a request-replay protocol, here the web browser client makes a request to web server which sends a reply message. For example when a user visits a website, a request is sent to the servers to download pages and images or

execute applications on the server, the server replays by sending the requested resource and the appropriate acknowledgements. Another important aspect of the Web is the Domain Name System (DNS). IP Addressing schemes are numerical and they are universal and unique. Domain Names provide a symbolic name suitable for human users and maps onto an IP address of hosts and networks.

1.6 Information Asset and Transaction Costs

According to [100] Information Assets including, expertise, trademarks, Market Intelligence, goodwill and corporate identity, will become for some organisations goods and services themselves. The customer loyalty card used by many retailers is prime example. Also publishers are creating corporate digital archives in order to extend the reusability of content.

The impact on the new economy is also assessed in [100], suggesting that many traditional business functions will merge. As more of these functions can be expressed as digital information then Metcalfe's Law can come in to effect.

The costs of carrying out business functions or transaction costs amount to the set of inefficiencies that add to the overall price of the ultimate good or service. These can include the cost of searching for buyers and sellers, learning about products and services, bargaining, and making decisions. Transaction cost where first properly described by Ronald Coase in 1937. According to Coase [100] commercial organisations exist because their transaction costs are cheaper than those involved when individuals conduct the same business with one another on the open market, therefore firms will expand to the point where the transaction costs become equal to the cost of carrying out the same transactions on the open market. As transaction costs add complexity, bureaucracy increases with the size of the firm. It follows that a firm should only conduct business functions that cannot be performed more cheaply by others in the open market. According to [100] the implications through digitization of both Moore's Law and Metcalfe's Law mean that many transaction costs are now lower and that there will more outsourcing and disintermediation, or cutting out of middlemen. The implication is that the relationships between different organisations will

therefore change and there will be processes of vertical and lateral integration.

1.7 Organisations

1.7.1 The cybernetic Organisations

The Cybernetic model of organisations is described in [95]. This model is based on flows of information going in and out of the organisation similar to self-regulating machines and organisms. This work, [95] has detailed accounts of empirical investigations and theoretical studies of organisations as systems. This particular view of an organisation is the result of social action feed back into the ongoing pattern of social relations, which is the realization that industrial organisations will have informal social structures that are systemic in nature. The Cybernetic Model is part of the socio-cultural system model of an organisation. This involves analytic structures as opposed to concrete ones which constitute membership groups or aggregates of individuals, identified by their function. The Cybernetic system is structured by the causal texture of interconnections that surround the enterprise creating processes transmitting, receiving and interpreting information, meanings and decisions. This can be seen as a progression from the *placid* randomized environment of perfect competition towards "*turbulent* fields characterised by complex and continuous interconnected changes" [95].

Cybernetics, from the Greek for "steersman" or "Governor", is an interdisciplinary science dealing with communication and control systems whether they are in living things, machines or organisations. First attributed to the American mathematician Norbert Wiener, Cybernetics is concerned with an information *feedback* mechanism for automation. In this context Information is regarded as statistical in nature, in a similar vane to Information Theory, which is concerned with the mathematics of communication, transmission and processing of information. According to Cybernetics the lowering of entropy, order, is less probable than a situation of rising entropy or chaos.

Industrial organisations and enterprises being treated as systems is part of general systems theory, with provides a generic framework of concepts and assumptions for use in the analysis of various phenomena, regardless of the particular scientific field. There is an organic analogy of open systems, in that there is exchange of material with the environment working towards a steady state or homeostasis where the system is as a whole remains constant regardless of fluctuations and failures with inputs and internal components. These *open systems* have equi-finality that is they will reorganize to greater heterogeneity and complexity, as opposed to a *closed system* with a maximum homogeneity of parts and relatively static equilibrium. Using the Biological analogy it must secure energy inputs from its environment in order to achieve its goals, such as surviving. This is the essence of the adaptive enterprise.

In regarding industrial organisations as systems suggests that it is more than an aggregation of independent parts, with a relatively clear boundary separating it from its environment, and that there is some overarching goal or set of values. To regard societies as having organic characteristics recognizes the systemic properties of social aggregates of different independent beings.

1.7.2 The Adaptive Enterprise

As we have seen the science of complexity includes industrial organisations as being within its scope. The *Firm* as an example of a complex adaptive system is found in [106], seeing the emergence of high-level order arising from the low-interactions of autonomous agents driven by simple rules or put another way people working together with in the organisation.

The Adaptive enterprise is defined in [113] as a firm that is responsive to changes in its environment, changing its behaviour, learning and evolving. It is the contention of the authors (The Cap Gemini Ernst & Young Center for Business Innovation) that being Adaptive is increasingly important in the current or recent global economic climate, which is becoming increasing volatile and more chaotic. They see change becoming more frequent, more sudden and less predictable. The Center for Business Innovation states their

aim, in this regard, as helping enterprises to thrive in a volatile economy by becoming permanently adaptive.

The causes of volatility, according to [113], are principally due to the networked and globalized economy. In world of increasing rates technological change and connectivity enterprises are facing greater uncertainty. Unpredictable change creates market conditions leading to pressure on costs and companies focusing on core businesses. In these conditions, the Biological analogy of the industrial organization is most applicable. The *Digital Ecosystems* proposed in [54] see these organic systems interacting, with a shift from competition towards more collaborative partnerships. Here the outsourcing of non-core business functionality is also advocated, leveraging an internet infrastructure to streamline interconnected internal and external processes. In this model internal assets are no longer essential with the focus on operational efficiency rather than ownership and internal control. Here and in [113] business boundaries can and should become more permeable with greater flows of information and transparency between organizations. These processes of lateral and vertical integration are defined in [96].

1.8 Systems Development

The IT industry has a reputation for the frequent failure of software projects, including both time and cost over-runs, teething problems are the expected norm for new systems. In [79] these issues are attributed to management and organisational issues, particularly in larger projects. Human factors in project outcomes are also dealt with in [86], where the code & fix approach is described. Most systems and software development is conducted without planned processes for control and monitoring. The processes are chaotic according to the Capability Maturity Model [79], for many projects there is not a stable environment so success is heavily dependent on key staff. The system is cobbled together from many short term decisions [86], growing in complexity, bugs become prevalent and it is increasingly difficult to add new features. According to [20] “the results bear certain similarity with to the construction of ambitious Gothic Cathedrals which from time to time collapse before they were finished.”

The low level rating on the Capability Maturity Model for many projects is attributed [79, 20] to a lack of appreciation that software development should be regarded as an engineering discipline. In [20] the author states “You will probably find no other branch of engineering involving a higher proportion of amateurishness than software development.” , and “too many people who have never been trained in computer science are entrusted with Software Development”. These sentiments can also be found in [79] and probably backed-up by other sources.

The adoption of development methodologies that impose discipline and planning controls similar to those in other engineering disciplines is a solution which is not always entirely without its own problems [86]. A reason for the failures of some methodologies is due to difficulties of defining fixed requirement for a software project in a world where business needs can change rapidly, so changes to in a specification of requirements are the norm. A new breed of Agile or lightweight methodologies is tackling these shortcomings. However as things stand many systems have been implemented which lack scalability and extensibility, have poor standards of design documentation as well as quality assurance, due to a lack of any methodology or testing regime.

1.8.1 Methods

The difficulties of schema integration resulting in the non-adaptive nature of autonomous and proprietary information systems have already been cited as motivation for this work. The burden here is upon the relevant knowledge workers too correlated and verify the relations and patterns within any distributed collection of information and data. Differences in user terminology were raised in [15] as a contribution to problems of schema integration. Human approaches to addressing this heterogeneity of abstraction and semantics of systems as well as requirements for scalability are considered here. Large and dynamic systems and the projects undertaken to develop then, will potentially grow in complexity, and as already discussed, many systems development life cycles collapse in chaos.

The art of programming itself, according to [11], is about managing complexity. Here the point is also made about projects failing due to complexity. Software Development is compared to engineering discipline in

[79], where standards are important. The importance of applying methodologies in software development is discussed in [20,79,84,86]. What the application of methodologies share with knowledge management strategies, particularly as facilitated through Web Services, is the need of stake-holding communities to share knowledge through communication and collaboration, this is a key aspect of e-business.

1.8.2 Objects and structure

Object Orientation is a widely used and understood paradigm, which is embraced in programming, software development, analysis and data management. Object Orientation is descried in [20] as Weltanschauug (World view). It follows ontological principles to provide holistic and human oriented objects that are problem-oriented abstractions, in what [16] describes as mimicking the human way of doing things. In [11] Object Orientation enables the use of the computer as an expressive medium, and points to the reusability and interchange-ability of objects, with the obvious benefits of reduced time to market and reduced complexity. Object Orientation replaced structural and procedural approaches in software. The essential difference is that in structured approaches the data and operations are separated, with Object Orientation the class concept of operations and data is combined behind an encapsulated interface. Object orientation recognizes software development is an evolutionary process [11, 20] where, unlike in structured methods, there is less of an assumption that a complex system will be constructed in one go, or single iteration. This is a systemic holistic thinking, which is regarded in [20] as the answer to the increasing complexity of software development, and problems of integration.

1.9 Data Management

1.9.1 Persistence

What elevates a computer from a calculator is the ability to save data, e.g. writing to disk, persistence data is assessable after the runtime execute of application, such as after the machine has been switch off, and is available

to other users. Persistence is one of the more important and obvious characteristics of a Database Management System.

A database is described in [70] as a collection of persistent data and the data has “more-or-less” independent existence or it is semi-permanent. We shall define a database as a structured and accessible collection of semi-permanent data. For instance a name and address on a particular organisations mailing list. Here we are regarding the database as a conceptual model for structuring a particular collection of data, information and ultimately knowledge that is presumably to be stored and used. The database management system (DBMS) is the technology for implementing one or mores databases, such as Access, Oracle etc. A DBMS creates an internal representation of the conceptual data model. The term often used to describe this structuring is schema. It provides a certain level of transparency over the mechanism of access and control over the details of reading and writing data to disk. This gives the users various means to view the data in a database and the ability to create, amend and delete data stored in a database, or transactions. Designing and building a database first involves modelling the data to design a schema, then configuring the DBMS and finally populating it with data, sitting down and typing or however.

1.9.2 Data Modelling

In [31] Data Modeling is described as “…organizing the required data structure into a form which can be represented by appropriate software” .By performance of data analysis a data structure or schema/metadata can be constructed for use in a database. Here in [31] three fundamental data models are described, namely hierarchic, network and relational. Arguably, a fourth approach is that of Object Orientation. Of these the most widely used data analysis paradigm is the relational data model attributed, to E.F. Codd. Relational Databases form the majority of commercial database management systems today.

The Relation model builds relationships between entities within a particular information space. Relational Data Analysis (RDA) is a constituent stage of SSADM (Structure Systems Analysis Design Methodology) and is described in

[Ashworth and Goodland, SSADM, practical approach] as well as in [31] and other texts [70]. RDA is concerned with the placing of data in tables. These Tables are also referred to as *relations.* These Entity-relations are comprised of rows and columns and represent particular entity sets. The rows of a data table represent the occurrences of a data entity or object, which are the subject of the data system being modelled. The columns in a relational data table represent a particular attribute of a data entity that is a property of that entity. The purpose of RDA is to organise data items into normalised relations to avoid unnecessary duplication of data items in different relations, and to avoid any anomalies occurring when updating the data. This process is known as normalisation.

Relationships between entities are bound through shared key attributes. These relationships can be of a *one-to-one* or a *one-to-many* (some times called *master-detail*) nature. The Relational model tackles the resolution of *many-to-many* relationships through the creation of an entity class from which an intersecting record can be created.

The Hierarchic model takes a users orientated approach by developing a structure from an obvious root to form a tree. IBM in their Information Management Systems (IMS) advocated this paradigm. In the Hierarchic model the use of pointers resolves the *two-parent* problem, reducing redundancy.

The Network data model, as proposed by CODASYL (Conference on Data Systems Languages, DBTG (Database task Group)), is concerned with modelling asymmetric network structures where there is no unique root to form a hierarchic *top-down* structure. The Network model relies on sets of Binary Relations to form building blocks of *one-to-many* relations or *owner-member* relations. Many-to-many situations are resolved by shared ownership of intersecting data, overlapping set types. According to [Bowers 88] the Network data model is amenable to flexible processing and manipulation of structure.

1.9.3 Data Management Systems

Already discussed are data modelling, and different DBMS that can be implemented to support those different models. Enjoying a critical mass of market support is Relational Database Base Management Systems (RDBMS). These databases are ideal for simple structured data types and will scale well with increasing data volumes and transactions. RDBMS was not designed for multimedia and is not suited for complex data types. Object Relational Database Management Systems (ORDBMS) add object orientated capabilities to relational databases enabling multi media capacities. The Objected Orientation (OO) philosophy is found in Objected Orientated database management systems (OODBMS). The systems enjoy niche markets in CAD/CAM and Telecommunications. Theses object oriented systems can deal with complex relations, multimedia and use optimizers to determine the best use of indices and layout. Also there the OO benefits in reduced time to market though greater flexibility.

The emergence of XML as data format has resulted in the prospect of a new type of databases that can exploit the advantages of extensible complex data structures and scalable schema. This complexity of structure however, presents new challenges and also the vast volumes of pre-existing relational data cannot be ignored.

A failure model for databases is presented in [70]. This includes multi-value problem, update insertion and deletion anomalies, the maintenance of relational integrity and the creation of views across tables. Some of the typical challenges encountered in the use of data management systems are discussed in [102]. These include redundancy, data control, interfaces, data integration, latency and the lack of reality modelled by systems. These issues are largely common to all data management systems and depend on the underlying systems, hardware and peopleware, the humans using and running the systems. Redundancy is simply the same data stored in different locations, the danger is that any amendments will not be carry across all instances of that piece of data, this was also a waste of disk space, not so precious now but once costly. Normalisation is the exercise of ensuring data is represented only once. In a relational model, schema is effective slimed down to the minimum entity-relationships. The lack of integration in legacy systems can be attributed limitations of the available

technology at time of development, and costs and disruption involved in solving these problems all at once.

As we have seen the DBMS is an integral component of a knowledge management and decision support systems. According to [102] data are un-summarized and unanalysed facts. Information is data that has been processed into a meaning form. Knowledge is the capacity to recognize what information is useful, and make use of that information, then being able to interpret that information and use it in decision making. Knowledge arises from patterns in information, and information from relations in data. The challenges are to use diverse data wherever it may be stored, through integration and then to tackle the peopleware through training and awareness.

1.9.4 JDBC

Applications such as DBMS often demand network connectivity and there usability and value is significantly enhanced by this, creating opportunities for multi-user, distributed and concurrent access. There are also possibilities for distributing and replicating data storage, and interconnecting different but conceptually related databases in multidatabases. Vendors of DBMS appreciate the value of using the internet for enabling remote use of a database, and most if not all major providers of database technology include internet accessibility to their products. Java Database Connectivity (JDBC) [5,94] goes beyond propriety mechanism to create an intermediate layer that is largely database and platform independent. In effect JDBC is a library of java classes that can interface with any RDBMS that supports open database connectivity (ODBC). This enables a programmer to interact with the RDBMS using SQL. Also JDBC provides extra functionality on top of the RDBMS such as processing data, caching derived results querying metadata and dynamically altering schema. Traditionally database implementations have been on a 2 tier client – server model, or the front-end/back-end concept. Where the client applications have required a detailed knowledge of the schema of the underlying database, by use of the DatabaseMetaData package found in the class library in JBDC, the Schema need not be hard coded into the client application.

1.9.5 JDO

Java Data Objects, found in the Sun One developer suite, [42] take things to higher level of conceptual abstraction, and transparent persistence/access, the programmer to handle even less details of access or even know SQL. Usually with Java programming objects are saved explicitly to an output stream, by writing fields to a flat file on a local disk, other wise JBDC is used to connect to RDBMS. The life-time of classes in Java is usually limited to the scope of the execution of the application or session. It is obviously desirable for objects to persist so the data of an object can be shared and reused. The approach of JDO brings the *plug-and-play* benefits of Object Orientated programming, such as simplicity reusability and encapsulations to applications using RDBMS.

1.9.6 XML and Database

The author in [60] poses the question, is XML a Database? An XML document may well constitute a structured collection of meaningful data, but it is not by itself a DBMS In [60] the answer is yes but only strictly speaking.

XML is also discussed in [125] with regard to its relevance to database management. The author dismisses XML as a suitable format, regarding it as a reinvention of hierarchic database management, which fell out of use, according to the author, because of a lack of an underlying data model, and forced “horribly complex” hierarchical relationships. The relational model was invented specifically to eliminate such hierarchal modelling. Specifically the relative weaknesses of XML compared to Relational systems [125] include:

- A lack of administrative functionality, interoperability and programmability and manageability.
- A lack of clear standards and the use of proprietary APIs and Query languages.
- Xpath does not support grouping, sorting and summarising.
- XQuery unlikely to support updates, inserts or deletions.

These difficulties according to [125] stem from the industries disregard for fundamentals. In the case of XML and Hierarchical models this would include graph Theory [50]. XML as a Database format is also considered in [60]

where the author approaches the issue in terms of what the format is suitable for. It is not suitable, according to the author, for systems involving many users, requiring strict data integrity and performance requirements. What XML is suitable for is configuration data files, personal data, content management systems and of course a portable data format. There are therefore reasons have a database use XML for its underlying data. Such a database that is especially designed to store and use XML documents is know as a Native XML Database [60]. A Native XML Database will have no prior need for configuration or mappings in order to handle XML documents, as its fundamental logical unit of data storage is an XML document.

Documents can be differentiated as either data-centric or document-centric, the data centric document uses XML as data transport for machine consumption, and are characterised by a relatively regular structure with little or no mixed content. The Document-centric has less regular, larger grained structure and more mixed content are designed primarily for human consumption.

Native XML Databases or most suited [60] for the document-centric, where there is a need to preserve processor instructors and comments. These databases also are useful when there is a specific need to support for XML query languages, and for data-centric applications using XML, for information interchange for example.

XML databases are particularly suited for use in content management systems. These are designed for managing documents for human consumption, such a website. Content Management Systems have in the background some sort of database and control aspects such as version control and access, and could include text editors and search engines. They allow for the separation of content from style with advantage in extensibility and also for integration of more data-centric document and legacy data residing in traditional databases.

Native XML databases have architectures that fall into two categories, text-based and model-based. The Text-based model stores data as text, this is either in a file system or large objects within a traditional RDBMS, and these are good at retrieval of predefined hierarchical trees. The model-based systems are built on the internal document object model, storing and

retrieving documents as DOM trees. The features of XML database includes the sort of thing found in any other DBMSs, such as transactional support, locking and concurrency. APIs based on SAX and DOM parsers provided OBDC-like interfaces. Native XML databases also utilises the concept of document collections, where the system can define a hierarchy of collections, spanning many documents. Querying, updates and deletions must also be supported along with indexing and the ability to deal with remote data.

These systems must also tackle the particular issues of round-tripping, normalization, referential integrity and scalability. Round-tripping refers to the process of storing then retrieving a document to see if the saved document is returned. This is important in document centric systems. Text-based modelled systems are exact in returning a document, whilst Model-based systems return a document only to level of their model. Issues of normalization, referential integrity and scalability require special consideration when dealing with XML.

Normalisation is less important to the document-centric systems. However for a XML database to implement a normalised schema, it is required to have linking mechanisms such as support for XLink. An advantage of XML over the rational model is multi-valued properties due to its hierarchical nature; parent elements can have many children.

Referential Integrity refers to the validity of pointers to related data, maintaining a consistent database. Again this will involve the use of a linking mechanism along with transactional and locking mechanisms.

Scalability will vary according to the implementation environment. The scalability of indexing technology will be same as other DBMS. Speed in the XML systems will suffer for the retrieval of the hierarchical trees different to ones written in the actual document, searching of un-indexed data will be also slow.

1.9.7 XML:DB Initiative

According the XML:DB initiative [133]to XML DB's are a opportunity for improvements in the management of data and metadata. The industry

initiative sees this new technology as lacking in specifications, which increases learning curves and prevents interoperability and adoption. Possible applications sited are corporate information portals, B2B document exchange and product catalogues. The goals of this interest community include:

- Technology Specifications for managing data stored in XML DB's.
- *Reference implementations* published under open source.
- Establishment of a community of interest, to exchange knowledge amongst parishioners and vendors.
- Evangelism of XML DB's raising visibility.

The Requirements of an API for XML DB's are set out in [136], these are to provide a wide usability, vender neutrality, programming language independence and make the must possible use of standards. Such an API should interface with SAX or DOM representations and give a textual representation of result sets. Functionality of the API should include insertions, updates, retrievals, queries, deletions and transactions. The XUpdate language [135] is presented as a solution.

The requirements and design principles of XUpdate are given in [134]. The overall objectives being to provide flexible query and update facilities to modify to contents of XML documents. The author states the language *must:*

- Describe how to query and update XML content.
- Be extensible.
- Formatted in XML.
- Operate on part of or the totality of one more documents.
- Provide query functionality.
- Include update mechanisms.
- Be able to insert new elements.
- And delete elements.

And Xupdate should:

- Be simple and straight-forward.
- Conform to XML Namespaces, XPath and XPointer standards.
- Include logical operators, such as 'and' and 'or'.
- Be able to perform numerical and date manipulation.

Xupdate is detailed in [135] this document specifies the syntax and semantics of the language which is expressed in the form of a XML

document. It makes use of XPath expression to select nodes for processing and querying. XUpdate instructions are themselves formatted in as XML documents. The features of the language include modifications, appending and insertion of elements and attributes, renaming and removal of elements and attributes, as well as addition and modification of comments and processing instructions.

1.9.8 Xindice

An example of an implementation of XML:DB API is Xindice [130], According to its makers is, design from the '*ground up'* to be a native XML database. The advantages of its use, cited in [130] include: no need for mapping to other data structures, and flexibility through the semi-structured nature of XML and its schema independent model. Being construed from the XML:DB API Xindice makes use of XPath and the XUpdate language. Language independent is also available through an RPC (remote procedure call) plug-in. The development of Xindice applications in Java is the subject of [131].

1.10 Intelligent Agents

Intelligent Agents are described in [78]; these software programmes carry out task on their owners behalf, using a degree of autonomy. Web Crawlers, spiders or robots of the web-based search engine are examples of Intelligent Agents. As well as this property of *Agency*, Software Agents which neither fit into a server based nor client-server model, have *mobility,* that is the ability to migrate between systems in a network. The Agent will require a degree of *Intelligence*, that is use Artificial Intelligence methods to carry out its tasks, applying application domain specific knowledge. These Agents can further be classified by processing strategy:

- *Reactive/reflective agents* Take action responding to events according to encoded conditions, there is no internal model. These agents will tend to have emergent behaviour.
- *Deliberative/Goal Directed Agents* use domain knowledge and planning in taking sequences of actions in the hope of achieving goals, these agents can proactively cooperate.

- *Collaborative* work together and can solve large problems that would be difficult or slow for a single agent, the results are the synergy of these autonomous agents. These Multi-Agent systems constitute a collection of autonomous software entities, with their own internal goals, here they can either have fixed goals or can change roles. The Agents in such a system will communicate with one another to achieve the collective goal.

The following BDI (Belief, Desires and Intentions) model is also used for Mobile Intelligent Agents:

- *Beliefs* constitute the Agents knowledge about the state of the world, or what it believes to be true regarding the state of the outside world. Although it can never have a complete knowledge of that outside world.
- *Desires* or goals are the assignments of goodness as to states of the world. On these the agent can reason about the state of the world.
- *Intentions* represent the course of action the Agent intends to take, these are its committed plans, based on its reasoning as to what it believes is the state of the world.

The benefits of using mobile agents are listed in [78]. These include reducing and balancing the load on communications and distributed systems, overcoming network latency. Also there is fault tolerance due to the autonomous nature of agents, they can keep working during network failure and can dynamically adapt to changes in system load.

Agents can undertake tasks in either a push or a pull mode, The *Push* or broadcast mode sends out updates, to web sites for example, whilst the pull mode retrieves information, this allows the user to explore the vast amounts of network available information. According to [78] intelligent agents will become useful personal assistants by searching, finding and filtering information. This involves the Agents using heuristic algorithms to search around the web. These filtering – seeking agents can also be classified as *Information Agents* whilst those acting as personal assistants can be classified as *Interface Agents*, these can be programmed to learn from the users input and feed back, follow explicit instructions or offer advice. Here this *software with attitude* should be enabling according to the authors, not frustrating or intrusive.

Problem solving through search algorithms using Java is discussed [48, 78], as are the benefits and considerations of using Java to construct Intelligent Agents. These include:

- *Autonomy*, where the Agent should run as a separate process or thread, which is supported by Java.
- *Intelligence,* Java has all base functions for behaviours and can hard code logic.
- *Mobility,* Java is portable and can use the Javabean delegation event model which allows dynamic registration of event sources and event listeners.

There is also a requirement to save the state of running process and resume possibly on a different system.

In [78] the authors set out the requirements for building Intelligent Agents. On the one hand they are seeking to create an easily understood illustration of an Intelligent Agent, which is however practical for realistic problems. The goals are seen as both application centric and agent centric. The application centric view sees agents as helpers, extending the applications functionality, where intelligent behaviour can be embedded in Java's Object Orientated Framework. In an Agent centric approach, The Agent monitors and drives applications. Here there is a requirement to for an Agent Management Application.

The required features of Intelligent Agents are that they must support a relatively sophisticated event-processing capability (if-then rules), backwards/forwards rule based processing and be able to add domain knowledge. Also the agent must have the ability to use learning algorithms, employ a messaging protocol and have persistence, that is be able to write to file.

Knowledge representation is an important feature of Intelligent Agents. In Multi-agent or Collaborative Agent systems Knowledge needs to be communicated, if this is done in a standards based manner then the implication is that these collaborating agents can be independently developed, in effect in an open multi-agent system. An example is KQML (Knowledge Query and Manipulation Language).

1.11 Related Work

1.11.1 Metacomputing

Decision support applications are a motivation of [1]. The aims of this work are the provision of scalable interface accessible to the widest range of users, who require access to large computational resources. Part of the DISCWorld project, the challenge is the seamless integration of those resources when spread over wide geographical areas, and can include several sources of data together with potentially demanding processing requirements. The key issues identified by the authors are the many of general issues surrounding distributed computing models covered in [43]. These issues include trust issues such as security, authorisation, access and monitoring, code performance and portability. Other performance factors including communications latency and bandwidth, acknowledgement of request, recovery and robustness, conflict resolution, priority and scheduling, adaptability, maintainability and configuration. The authors also note that users are likely to access resources through widely varying network bandwidths with differing powered computers. The solutions too many of these problems can by found, according the authors, by the adoption of the server technology of the World Wide Web.

In [1] the authors see Java as the natural choice, due to the in built networking and platform independence, by using Java enabled web browsers or JVM (Java Virtual Machines) installed on client machines. Java is also seen as suited for applications providing services. Here native code and optimization can be used instead of the higher level portable byte-code. Server can themselves execute other service on remote machines, making it possible to distribute calculations over a number of server machines. A characteristic of DISCWorld is this network of peer to peer servers. Essentially DISCWorld has decentralized architectural scalability through Server-less smart-middleware software glue, [2] in which applications are embedded in a semi-opaque cloud of services and resources. Replication and mobility of services, resources and storage are achieved by the use of symmetrical nodes on a computational grid in which there is no intrinsic hierarchy.

Approaches taken by DISCWorld to data storage are discussed in [5,28,8,9]. In [8] legacy support for applications is discussed. Here DWorFS (DISCWorld File System) is used to interface heterogeneous collections of distributed storage resources, using an underlying NFS (network file system). Issues of integration and interpretation of heterogeneous multidatabases are discussed in [15]. Mechanisms for manipulating and browsing distributed data archives are also discussed in [5] and in [8]. The challenges of interfacing and specifying required data are also addressed, [8] points to possibilities of Multi-component look-up and pre-fetching of data. Discussed in [5] is a number of suggestions for future directions, including the use of mobile code and data, as well as *value adding* and caching of intermediate results.

The use of flexible access to storage resource is discussed in [28] where a distributed presence can be created for data at all nodes in the distributed system, using DRAC (Distributed Active Resource Architecture). This provides a means of abstraction over data resources. DRAC relies on mechanisms for data resources to exchange information on each other. The use of rich-data pointers for pre-fetching of data or Data Futures is discussed in [9]. Here DISCWorld Remote Access Mechanisms (DRAMs) are described. Data Futures allows means of addressing data products that have yet to be created. The Specification and management of complex processing tasks is the subject of [7] where directed a-cyclic graphs are used to represent operators, expressed as scripts in XML based DJPL (Distributed Job Placement Language).

The Naming or the means of addressing services and resources in scalable and dynamic environs is recognised as a problem by DISCWorld [9,7].

Replication and consistency between computation nodes in a distributed system are discussed in [28,6], “Gossip” architectures are used, where nodes share information about one another. Flexible update propagation methods are detailed in [28, 6], in which *Bayou* an anti-entropy protocol is described. A Gossip Protocol for resource discovery information interchange is used in [6]. In this model, tasks and services are linked in a Task Graph. Here a percolation problem is posed by a global graph of independent resources, and the challenge is to maintain connectedness whilst minimising cached storage at each node.

An Issue for DISCWorld is trust and security, LDAP (Light-weight Directory Access protocol) is suggested in [6,7]. This could provide for access polices, also techniques described in [43], such as digital signatures, certificates, and SSL's (secure socket layers) could be applied where appropriate.

1.11.2 Knowledge Management

Other motivational issues are addressed in [Albert, 3,4,10]. In [10] the need for remote access is also discussed. This is a consequence of the dependence and cooperation between organizations and that of departmental autonomy. This results in the need to interconnect existing heterogeneous information systems. The aim of this is to address the problems of schema integration and introduce a modeling formalizations incorporating use case grammar for schema comparisons, conformations and merging. The author proposes the use of conceptual graph theory as the basis for a conceptual modeling language representing knowledge structures.

In [4] the motivation is the creation of corporate memory. This paper examines the use of XML in knowledge management, and uses enterprise guided ontologies for searching XML documents. It defines corporate memory as an explicit, disembodied, persistent representation of knowledge and information in an organization, in order to facilitate its access and reuse by members of the organization.

The work of [3] is motivated by the limitations of keyword based search engines.
Here RDF (Resource Description Framework) is exploited for expressing and interpreting metadata having been mapped into conceptual graphs. This has as it aims the building of the semantic Web.

The key role of information management in heterogeneous metadatabase is examined in [103] Here the author points out that organization typically own distributed databases of different types. The author also recognizes the potential of the web to provide universal access to countless sources of data, hence the need for schema integration.

[22] Presents algebra for operations on semistructured Data, based on the concepts of Ontology composition and articulation between ontologies. Here

the Algebra constructs connected directed labelled graphs to represent relationships between data, this is applied to Web Documents. The Algebra uses wrappers to mediate semantic content. Uses include topic and schema discovery.

The conceptual architecture for heterogeneous information spaces is introduced in [15] where servers and system components are distributed across system boundaries. The model described information elicitation, which provides for serendipity, explorations and conceptualisation. Here two phases are outlined regarding user interaction with a multi-database, they are schema discovery and the transaction phases.

The author of [75] investigates the implementation of a structure document database with much of functionality of database management systems. This relatively early project utilises SGML (Standard General Mark-up Language) the forerunner to XML and making use of the hidden information.

The design approach of structured document databases can be categorized under two headings [75], Top-down or bottom-up.

Top Down

In This design approach, design starts at a conceptual level, creating a model of the database upon which the design of operations is based. Eventually this leads to a low level implementation of that model. There are two sub-categories of the top-down approach, include a complex Object approach, in which queries are specified in an algebraic form, reflecting the complex-object nature of documents. The other approach is a Grammar-based approach, using Content Free Grammars (CFGs) to describe the database schema. Making use of parse trees of the grammar the data model uses ordered tuples and sets. The use of object orientated databases is a variation on the complex object approach.

Bottom up

With a bottom-up approach, an implementation strategy is first designed, with a data structure for indexing the data with operations created to efficiently use that data structure. The use of indexing strategies can be found within the design of search engines.

1.11.3 Graph Drawing

The problems of graph drawing are described in [32] as a combination of computational geometry and graph theory with applications in networks and software visualization. Here a model is presented for dynamic graph drawings, based on performing queries and updated on an implicit representation of the drawing. Covered by in this paper are algorithms for drawing trees, series parallel and planar graphs.

The VLGD algorithm is presented in [72] this paper addresses the issue involved in drawing large graphs includes fish-eye, zooming and clustering techniques.

The authors of [73] use iterative heuristics in layout adjustment algorithms in graph drawing. The objective is to retain the overall shape to preserve the users overall mental picture, using a process of *cluster busting.* In [74] efforts are described to graphically represent a dynamic portion of the WWW as a graph drawing where pages represent vertices and the hypertext links as directed edges.

1.12 Conclusions

In the changing world of the information age, organisations need to adopt new strategies to cope with the proverbial economic rollercoaster. Whether to identify these strategies or to manage change itself, the systems managing that organizational knowledge will increasingly need to be decentralized and collaborative as well as scalable and extensible. With these requirements those systems become more complex and burdensome to manage and administer. The challenge is to develop the tools and techniques to enshrine extensibility into the knowledge management systems, and to allow bottom-up design and development to occur in a manageable and grounded way.

This will require trust and consensus across distributed systems, in order to interpret and exploit the combined informational wealth found within those systems. Traditionally this requires a detailed human understanding of the

underlying data schema. Discovering and conceptualing the structure and properties of information systems is intrinsically difficult, particular for large systems that are subject to arbitrary change. Yet schema knowledge is as vital for interpreting and analysing data content as it is for the integration of different systems.

If schema and metadata can be discovered then shared with others, securely and accurately, with out central controls, then individuals and organisations can effectively share and incorporate each others knowledge.

Mechanisms are needed, to supplement and support the human management of these large and distributed data systems, in particular, the ability to monitor changes in schema and highlight changing content. This will require heuristic work flow applications for semantic web building that can either function as daemons, running as a continuous background service, or as a user initiated operations. The challenges involved in implementing such mechanisms, firstly involve being able to scan potentially large and distributed stores of data in a timely and efficient manner and then to be able to represent and display those discoveries. The solutions to these challenges can potentially be found in graph theory and technology can be found within the design of software agents and Web crawlers in the search engines used in the web. Here the crawl, covering a targeted region of cyberspace, such as a collection of servers and resources, would dynamically build and maintain a database storing metadata and schema, in other words data on databases, what could be called a metadatabase. This could then be used to create a common view of different internal representations of conceptual data sets from a common domain. A graphical user interface then could facilitate visualization, interaction and management over potentially heterogeneous and independently developed systems.

2 Glasnost

2.1 Openness

Already discussed have been the problems common to software development and the role of methodologies. Also addressed is the need for acknowledged standards for the creation of open distributed systems such as the Internet and Web. The concept of *Openness* has several possible definitions depending on context. In [14] it is suggested that Openness as applied Web Services, should mean both Open Standards and Open Languages.

Open distributed systems are ones [43] that support resource sharing and are extensible in terms of hardware and software, which is a system, which can be extend and re-implemented in various way. This can be achieved by publishing key software interfaces to components to facilitate a vendor neutral and platform agnostic mechanism or protocols for communicating across a heterogeneous infrastructure.

Heterogeneity is the term applied to the variety of computers, networks and associated software that can be encountered. The variation will include different implementations of Internet Protocols (IP) over different networks, differences in the representation of data, such as the Byte ordering of integers, and different operating systems, such as Windows, UNIX, and Mac OS etc. For open distributed system to overcome the problems of heterogeneity, different operating systems will need to access the same IP interface, and standards for messaging need to be agreed by developers of different applications.

Protocols are well known sets of rules and formats used for communications. They define the sequence of messages to be exchanged and the format of data within those messages. The publishing of such protocols proceeds through the use of Request for comments (RFC) documents; this bypasses the cumbersome and slow procedures of official standardization. RFC are now the basis of all Internet documentation.

Open Distributed systems such as the Internet are necessarily scalable and extensible [43]. The scalability of systems is a requirement to remain effective when there is significant increase in both resources and users. This involves performance issues and allocation of memory space for Internet addresses. The requirement is that the availability of resources remains proportional to use and the accessibility of data remains manageable. Also functionality should be added to and increased transparently such that new features can be opted into without disruption to existing client application functionality. The Internet has successfully grown in size and capacity and has managed to adopt new protocols such as HTTP (the Web). This has occurred as a consequence of industry partnerships and consortium with little overall governance.

The Internet and Webs success is in spite of the proprietary nature of most software and the systems development life cycle, driven by the commercial interest of vendors and end users alike. Collaboration and extensibility are essential features of the new economic business models. We have considered the ownership issues of the opens source debate and agile adaptive methodologies [86] and their relationship to the Capability Maturity Model and Total Quality Management, in meeting the requirements of adaptability and scalability.

2.2 XML

Standards based computing is typified and currently dominated by Extensible markup language (XML), this is a subset of Standard Generalised Markup Language (SGML) as is HTML. A markup language can contain text and multimedia elements, these elements define chunks of a document and any special characters along with attributes that format and enhance those elements. SGML has its origins in a 1986 ISO standard. It was created for document publishing and distribution [18], and uses Document Type Declarations (DTD), which specifies the rules for a documents or set of documents markup language.

The generality of SGML made it complex [18], XML is a less complex way of developing markup languages. The need for a way to create domain-specific markup languages or a vocabularies, this goes beyond the fixed syntax and semantics of HTML, where the tags only specify display attributes, in XML tags can be use to specify meaning. The features of XML include platform and device independence,

support for DTD's, extensible structure, and text encoding such as Unicode. The use of DTD's is optional. An XML document can be processed by either client or server applications, allowing for the optimization and distribution of resources. It also means that presentation and content of information can be separated.

The following is an example of a simple XML document; it describes a collection of images.

```
<?xml version="1.0"?>
<PHOTOS>
 <PHOTO>
  <NAME>Arab03</NAME>
  <FILE>arab03.jpg</FILE>
  <CAPTION>an arab</CAPTION>
  <PATH>Arab03.jpg</PATH>
 </PHOTO>
 <PHOTO>
  <NAME>Two Horses</NAME>
  <FILE>horses01.jpg</FILE>
  <CAPTION>Bow Rockers</CAPTION>
  <PATH>horses01.jpg</PATH>
 </PHOTO>
 <DIR>images</DIR>
</PHOTOS>
```

The first line <?xml version="1.0"?> is referred to as the document prolog, this is followed by the what is called the document instance. The top level element <PHOTOS> is enclosed in by tags "<>" the end tag is started by "</" . The document has a hierarchal trees structure where <PHOTO> is a child of <PHOTOS> which is therefore is its parent , <NAME> and is a child of <PHOTO>, <NAME> and <FILE> are siblings. This example includes no reference to a DTD, would be denoted by the key word, "!DOCTYPE". An example of an element with an attribute could be:

```
<FILE type="jpeg"> horses01.jpg</FILE>
```

An empty attribute would be denoted

<CAPTION/>

The structure is extensible as elements can be arbitrarily inserted or excluded. An XML document can be presented by the use of cascading style sheets (CSS) or XSLT (extensible style language transformation) which convert the document into HTML. The following XSLT document converts the above example to a HTML table also show.

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
 <html>
 <body>
  <table border="2" bgcolor="aqua">
   <xsl:for-each select="PHOTOS/PHOTO">
    <tr>
     <td><xsl:value-of select="NAME"/></td>
     <td><xsl:value-of select="FILE"/></td>
     <td><xsl:value-of select="PATH"/></td>
    </tr>
   </xsl:for-each>
  </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

Arab03	arab03.jpg	Arab03.jpg
Two Horses	horses01.jpg	horses01.jpg

This requires an XSL compatible browser, in this case IE explorer, or XSLT engine such as Xalan.

2.2.1 XML Standards

XML and other web standards and recommendations are under the stewardship of the World Wide Web consortium (W3C). Standards designed to work in conjunction with XML include the XLink, XPointer, XPath and XSL used in XSLT. All described in [18]. Xlink (eXtended links) and XPointers (eXtented pointers) are concerned with linking mechanism in XML documents. XPath expression are used to specify the path to particular nodes of a document. XPath support XPointers and XSLT.

2.2.2 XML API's

2.2.2.1 DOM

Applications Programming Interface's (API's) for XML, fall into two categories, trees based and event based. An example of an event based API is SAX (Simple API for XML) [40]. The Tree Based API is typified by DOM (Document Object model). The DOM defines the logical structure of documents and presents then as a hierarchy of node objects, which may have child nodes. This API allows for the building of internal representation of documents enabling navigation and interaction. It is not however a binary specification or a way of persistence, it specifies interfaces not data structures. The DOM approach has disadvantages as it can put a strain on system resources for large documents, round-tripping can be incomplete, that is documents returned for a DOM may be different from the original parsed document, the return document will be to the same level of hierarchal detail as the DOM tree, which not have the same depth as original.

2.2.2.2 SAX

Event based API such as SAX work by reporting parsed events directly to the using application. It does not build an internal tree or other data structures. The application developer can construct their own data structures using call back event handlers. SAX is not an XML parser it is a common interface implemented for different XML parsers. Examples of Java applications using SAX are given in [40]. An example of an XML parse is Xerces from the Apache Foundation [132].

2.2.3 Java XML API's

A description of API's is available from Sun Microsystems for Java, and their use is given in [41]. This make use of both the SAX and DOM approaches as does Xerces. The API's in [41] are categorised as either document orientated or procedure orientated. The former includes:

- JAXP (Java APT for XML processing) this uses parses to process XML documents.
- JAXB (Java Architecture for XML binding) binds XML elements to classes.

The latter, procedure oriented includes:

- JAXM (Java API for messaging) used for SOAP messages.
- JAXR (Java API fro registries) for access to business information registries.
- JAX-RPC, this allows remote procedure calls using SOAP to access remote servers.

2.2.4 Vocabularies and Schema

A vocabulary is according to [19] a list of terms used in communications. XML vocabularies can be defined by the use of a DTD, which provides a schema of a document or collections of documents. A Schema provides documentation, validation and can be used in code generation. A DTD is a subset of a pre-existing SGML standard, and has limitations [19], in that it cannot easily combine different vocabularies. XML schema, formatted in XML, and denoted with the file extension ".XSD" includes a rich data definition language to improve the precision of a vocabulary and the validation of documents, it also a better conceptual match to class inheritance. A namespace identifies the location of a DTD or XML schema, given by the keyword xmlns. The following document combines two vocabularies, namely Resource Description Framework (RDF) and Friend of a Friend (FOAF). The use of a namespace allows an independent third party to understand the terms and semantics being used.

```
?xml version="1.0"?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/">
<foaf:Person>
    <foaf:name>
```

```
      <foaf:firstName>Duncan</foaf:firstName>
      <foaf:surname>Stonebridge</foaf:surname>
    </foaf:name>
    <foaf:mbox rdf:resource="mailto:duncan@softengines" />
    <foaf:homepage rdf:resource="http://www.softengines.co.uk/" />
<foaf:logo  rdf:resource="http://duncanni.users.btopenworld.com/images/logo.jpg"/>
<foaf:project>
  <foaf:document rdf:resource="http://duncanni.users.btopenworld.com/project.html"/>
</foaf:project>
<foaf:organization>Softengines</foaf:organization></foaf:Person>
</rdf:RDF>
```

By itself an XML is not a technology, it is a way of formatting, structuring and describing data either in transient message or a persistent document stored in a file system. Being based on open standards it gives different developers a common medium, a lingua franca, in which their applications can write data, allowing for the sharing of data across different systems and administrative domains.

2.3 Open Source

The current open source movement and associated debate are discussed in [79, 88]. The movement was founded by Richard Stallman, a computer scientist at MIT, who in 1984 established the Free Software Foundation. The aims of the organisation are to create high quality freely available software, the idea being that it would promote the free-flow of ideas in which programmers could learn from each, and by debugging each others software ultimately create better quality software. New software, it is argued, is often best developed by adapting existing software, In copying standard features the overall feel and usability is standardized with learning curve advantages.

The arguments against this position are that it removes the financial rewards for developing software. These arguments are countered by pointing out that financial rewards are not the only incentive. Programmers stand to gain kudos by making successful contributions to popular or successful software, and besides many essentially free open source software products are bundled with paid support

contracts and services. Programmers involved in developing open source software can sell their skills on consultancy basis, delivering training and customising their software for end users.

There are issues of version control where bug fixes and modifications can introduce errors. Also the value of software goes beyond the cost of its development where the details of the source code of a corporate application may form part of the core business processes representing the enterprises market differentiation and competitive edge and therefore will be kept secret.

Open source is regarded as a methodology in [86]. The author describes the open source community as having process geared towards distributed and highly parallelizable debugging. With open source development there is no overall or centralized ownership. In [88] the General Public license (GPL or Copyleft) and implications and application beyond software are discussed. Copyleft stipulates that the particular work can be copied and reworked in a way as long as the modified work is also published under copyleft, thus preventing the work from being co-opted into later proprietary products and thus will remain free.

The open source debate is described in [88] as a “political battle over the ownership of knowledge”. Here the issues are compared to wider debates of corporate power and globalisation where Open source can be regarded a libertarian alternative.

2.4 The Semantic Web

The Web of today is a “universe of network-accessible information”, according to [59], which has proved its worth as a medium for human-to-human communication. Currently this universe of information is primarily for human consumption. Whilst the web is machine-readable as well as human-readable, its contents can largely be only understood by humans, before you enter into the realms of Artificial Intelligence. The aims of the semantic web are therefore to make the web’s content more machine-understandable, through the use of self-describing information or metadata, from which ontologies can be constructed. Ontologies are machine-understandable vocabularies containing definitions and taxonomies of terms in which knowledge can be represented and shared. The Semantic Web will provide a medium for software agents as well as humans to discover and share meaningful knowledge.

The vision of the Semantic Web is set out in [59,107,115,116,117, 118] by Tim Berners-Lee, largely credited with the invention of the Web (see The Web, History). A high level overview of the Web is taken in [59], where the creator describes the design decision involved in creating the web. Here and in [107] the Resource Description Framework are introduced. The objectives of [115] are to set out an overall plan for the implementation of a set of connected applications for data on the web as to form a consistent logical semantic web of data. In [59] and [107] it states the goal of the Web to be "useful not only for human-human communication, but also that machines would be able to participate and help" [107]. This machine understandable Semantic Web is compared in [56,115] to a Global Database, direct comparisons with the ER data model are discussed in [115,116]. A Key Element in the creation of the Semantic web is the use of RDF, the specification of the syntax and vocabulary can be found in [107]. Predating the Semantic Web Road Map [115] is a paper by the same creator [116] discussing Metadata Architecture, both works according to the Creator introduce the concept of machine readable data and self describing information on the Web. In [116] axioms of Web Architecture are presented by the creator, "Web Architecture from 50,000 feet" [59], is according to the creator a high level overview of the architecture of the web, "Necessarily, from 50,000 feet, large thing seem to get a small mention". Here the Web is defined as a universe of network-accessible information, its goals and core design values are outlined by the creator, as well as the duality of human and machine readability.

2.4.1 Metadata

Metadata is defined by [116] in the context of the semantic Web, as machine understandable information about Web resources or other things. Here axioms of Metadata Architecture are described. The first states that *Metadata is data*, this is expanded upon by the creator who describes how Metadata about one document can occur either within that document or within a different separate document, in other words Metadata can be a first class object. It follows that the first axiom has a second part, *Metadata can describe Metadata.* The second axiom concerns the form of Metadata; it states that *the architecture is of metadata represented as a set of independent assertions.* Assertions about resources are often referred to as attributes of the resource; an appropriate space in which to define attribute names is

the URI space. The goal of Metadata Architecture set out in [116] is to make available, as far as possible, the syntax and semantics of a resource through referencing a Metadata document.

In [5] Metadata is described as often being used in two sense, firstly it is used to refer to contextual or auxiliary information, and secondly to refer to the schema of a database. In the context of [5] auxiliary information is defined as extra information associated with a resource, such as file name, file size etc. about data. In [43] Metadata is described in the contexts of file management systems, such as file attributes and directories, whilst in [19] Metadata is the annotation within the tags of a document. Either way, whether the information or data describes schematic structural details or auxiliary contextual information, it is still information about information and data about data.

2.4.2 Resource Description Framework

An introduction to and examples of Resource Description Framework (RDF) are found in [107,120]. RDF provides a framework for describing and interchanging Metadata. It is machine and human readable and can be expressed in XML, it is used to describe resources without making assumptions about a particular domain nor defines a priori of the semantics.

The basic RDF statement consists of a three objects, a Resource (identified by an identifier such as a URI), a Property of that Resources/URI and the Statement or value of that Property. This Semantic Web building block can be regarded in terms of subject (resource), predicate (property) and object (literal/statement) where the literal/statement can be resources [107]. The following is an example of a section RDF document, it is generated from the references used in this work. The resource "DHC020" has property a "Title" with the value "DISCWorld: A Distributed High Performance Computing Environment", this resource could be identified by a public URI.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Description  rdf:about="DHPC017">
<title>World Wide Web Server Technology and Interfaces for Distributed, High-Performance Computing Systems</title>
<date>August 1997</date>
```

```
  <author>Silis, A</author>
  <author>Hawick, K</author>
</rdf:Description>
<rdf:Description rdf:about="DHPC020">
<title>DISCWorld: A Distributed High Performance Computing Environment</
title>
<date>November 1997</date>
  <author>Patten, C</author>
</rdf:Description>
<rdf:Description rdf:about="MyReference#3">
<title>A Conceptual Graph Model for W3C Resource Description Framework</title>
<date>1999</date>
  <author>Cordy, O</author>
  <author>Dieng, R</author>
  <author>Hebert, C</author>
</rdf:Description>
</rdf:RDF>
```

RDF can support containers for repeat properties, including; Bag, for unordered lists, SEQ, for explicit ordering of list and ALT for lists of synonyms for a particular descriptor.

RDF has the following characteristics [120]. Independence, as an open standard, and can be interchanged as it can utilise XML. It has scalability and an extensible data model and can therefore grow with the web. Properties can have their own resources like any other resources, values can be resources, and statements can be resources too. An RDF statement can be mapped directly to conceptual graphs [3], and provide an opening for ER-modelling on to the web [116]; it can be used to represent nodes and relationships. RDF is ideal for knowledge representation and therefore can be used to build digital libraries, collect resources into logical documents, and be used to represent intellectual property and digital signatures [59]. These metadata applications can then be accessed and used by intelligence agents in resource discovery, knowledge sharing and exchange.

2.4.3 Dublin Core

The Dublin Core Metadata Initiative (DCMI), named after the original 1995 workshop held in Dublin Ohio, is an interest community committed to the building of metadata standards [119]. These standards, practices and polices are seen as important to the emerging infrastructure of the internet. The initiative believes that there is a need to adopt a common core of semantics for resource description. Oversee by a board of trustees, the initiative is dedicated to the promotion and adoption of metadata standards and the development of a metadata vocabulary for describing resources, enabling metadata discovery by software agents and search engines. The Dublin Core Vocabulary can be expressed within RDF statements. The following is the earlier bibliography fragment using the Dublin Core specified metadata elements, date, title, creator etc.

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
<rdf:Description rdf:about="DHPC017">
<dc:title>World Wide Web Server Technology and Interfaces for Distributed, High-Performance
Computing Systems</dc:title>
<dc:date>August 1997</dc:date>
  <dc:creator>Silis, A</dc:creator>
  <dc:creator>Hawick, K</dc:creator>
</rdf:Description>
<rdf:Description rdf:about="DHPC020">
<dc:title>DISCWorld: A Distributed High Performance Computing Environment</dc:title>
<dc:date>November 1997</dc:date>
  <dc:creator>Patten, C</dc:creator>
</rdf:Description>
<rdf:Description rdf:about="MyReference#3">
<dc:title>A Conceptual Graph Model for W3C Resource Description Framework</dc:title>
<dc:date/>
  <dc:creator>Cordy, O</dc:creator>
  <dc:creator>Dieng, R</dc:creator>
  <dc:creator>Hebert, C</dc:creator>
</rdf:Description>
```

</rdf:RDF>

Here the Dublin core elements are within the RDF description, are properties (i.e. <dc:date>November 1997</dc:date>) of the resource identified by the about attribute (i.e. <rdf:Description rdf:about="DHPC020">). The use of RDF in combination with Dublin Core allows for meta-metadata and repeated properties, as RDF can disambiguate the meaning of repetitions. The vocabulary also includes elements for version control and identifying relations between resources.

2.4.4 Ontology

As with the concept of web services the work into ontologies according to [51], is driven by eBusiness needs. The dictionary defines the term ontology as being a branch of metaphysics concerned with the nature of being. In the context of ontological engineering it is a basis for understanding the conceptual structures in the world around us, helping us represent and communicating knowledge and providing a basis to construct vocabularies for particular topics and domains. A vocabulary is the collection of terms used in communication [19].

The role of ontologies in the B2B models is discussed in [51], the objective of applications to meaningfully share information is seen as a more important factor in creating electronic commerce than issues of reliability and security. The objective is to create standards for interoperation, to allow the formation of large and sustainable electronic trading groups. Previously this would have been hampered by a lack of a common language or format, multiple point-to-point interactions, and multiple, disparate catalogues, all supported by multiple levels of technology and proprietary tools.

Here the notion of the semantic web is addressed, by describing the current generation of websites as being designed and developed for interaction by human visitors, whereas the growth in e-business models means websites will have to interact with other systems. These interactions could include distributed problem solving and knowledge sharing. Commerce Agents are introduced in [51], these software components for complex interactions will require greater levels of semantic content to be made explicit and represented, hence the need for shared open

ontologies. Where ontologies can be open and shared then different systems can share knowledge.

2.5 The Promise of Web Services

Web Services are heralded in [36] as the most significant change for 40 years, within the IT industry. The business drivers of the approach include the need to integrate disparate applications distributed over the internet. Web Services can be described as a set of standards that enable distributed applications to "find each other and to talk to each other". In [85, 12] Web services are described as programmable resources that are accessible via open internet protocols. Here loosely couple components of a distributed computing environment use a messaging architecture based on the open standards provided by Internet communication, providing a *neutral glue* [12] to bind heterogeneous systems to a single framework. With *Loosely coupled* integration much of the nuances of tightly coupled integration are removed. The Framework of Web services is built upon the Internet foundation of TCP/IP and the HTTP communications protocol which enabled the use of Mark-up languages, client-side processing, using scripts and applets and server-side processing to delivers dynamic content and execute server transactions.

The Web has always been about open systems and platform independence where the responsibility is upon vendors to comply with known and published standards for information interchange. In [14] *Openness* and the dangers of vendor driven standards are discussed; open standards must develop in consultation on an industry wide basis. The natural choice to construct these new standards is XML, because of its suitability for information interchange in open systems. According to [19] the openness and extensibility of XML vocabularies is equally applicable to the both content and structure of the message protocol. SOAP or Simple Object Access Protocol is discussed in [19, 36]. The proliferation of custom defined XML vocabularies and competing standards is highlighted in [37]. Here the emergence of supersets of schema is discussed.

In [85] it is suggested that web services is a solution to interoperability problems faced by heterogeneous IT infrastructures, and that the approach will lend itself to

collaborative business and organisational partnerships, facilitating the Business Process Outsourcing in *digital ecosystems* [54] and convergence of stakeholder communities [19], by allowing organisation to focus on core competencies and service delivery [12, 19]. The approach also provides reusability, flexibility and "lower cost of ownership" reducing time to market and increases return on investment, without the need for additional technology or re-investment in hardware [12]. A major concern with any distributed system is security, with Web Service tried and tested Internet security measures, such as SSL encryption, can be bolted on to any application.

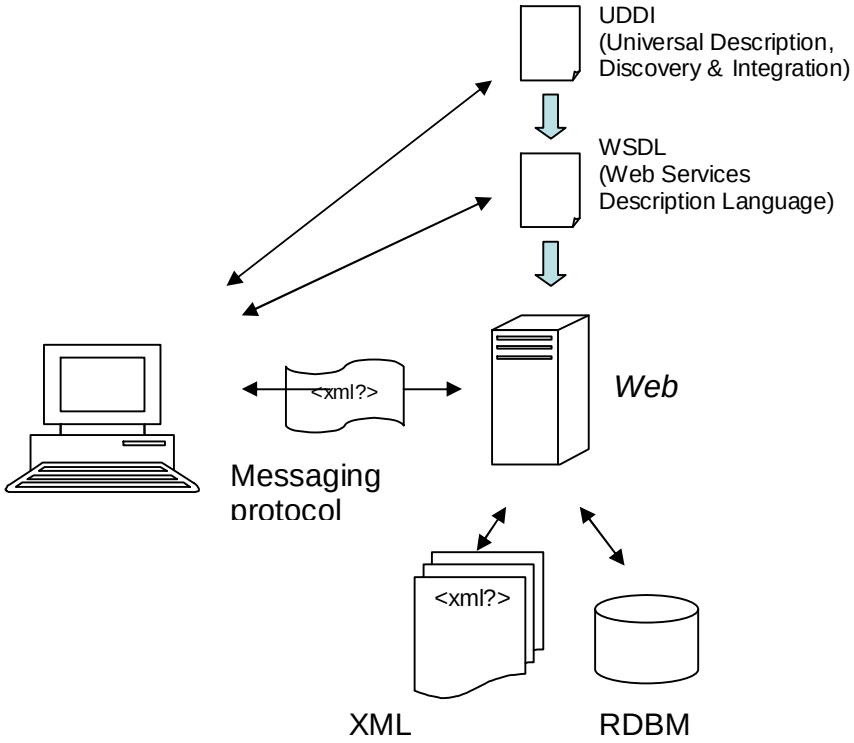
Clearly Web Services is Object-orientated in approach through the use of loosely coupled components. "There are similarities that encourage comparison with component-based development architectures and it offers the benefits associated with object-orientation - all built on an open systems platform" [85]. The implication is that methodologies and *open languages* [14] are a consideration when developing Web Services.

The potential economic impact of web services is discussed in [36], the current pre-web services IT industry is compared to a cottage industry where skilled practitioners duplicate their efforts to the cost of the individual organisation. This duplication of IT services is seen as being highly inefficient. The issues over outsourcing are raised by the author who sees web services as a middle ground between entirely surrendering control of systems to a third party and the in house alternatives. Splitting systems between internal management and external bodies has traditionally caused interface problems. Web Services is a technology that provides a bridge between these approaches. This will allow the organisations to concentrate on implementing business logic rather than technical issues of interoperation. The author sees this as reducing transaction costs to an organisation and could result in the shrinkage of internal IT departments, and the emergence of service providers that will be able to tap into and reuse the considerable investment there has been in the industry.

2.5.1 E-Business Architecture

The requirements for E-Business Architecture are outlined in [19]; these include Just-in-time integration, rapid time-to-market and dynamic configuration and reconfiguration of business models. The author also described the need for web

based information and loosely coupled services, minimal coordination and control of service end points as well as stating the need for platform and language independence. Such requirements, according to the author, point to use of HTTP, being a ubiquitous protocol for transporting information. He states “This broad-based integration, including inter-enterprise system integration has proven to be difficult in the loosely coupled chaos of incompatible platforms and information that are a common reality”. The issues for deploying web services are discussed in [19]. There are three aspects to this architecture. First is messaging protocols, such as SOAP, then there is the need for the services to be self describing and to be able to discover one another over the internet. According to the author XML’s verbose syntax may not be best suited for high performance systems, but be more applicable to non-real-time integration on B2B eCommerce. This is due to XML verbose syntax, an observation also found in [60]. However Web Service Architecture does herald the promise of Just-in-time integration and dynamic configuration. Standards for web service description are embodied in WSDL, or Web Service Description Language [110]. The goal of ubiquitous description and messaging for globally distributed applications can according to [19], be aided by the work of [111] UDDI or Universal Description, Discovery and Integration. The goal of UDDI is to be a key component in enabling enterprises to quickly and dynamically discover and invoke web service both external and internally[111] . This will involve a registry of business entities and contact information, from which a list of services can be requested and then return in a WSDL document. The following diagram illustrates this.



2.5.2 Technology Choices

The choices of tools and techniques required for the construction and development of the proposed systems for knowledge sharing will be determined by the requirement to interoperate, or function seamlessly across heterogeneous systems, and therefore have the qualities of openness. The benefits of using Web technologies are clear, either using the classic client-server model or other peer-to-peer models. In [14] the author concludes that methodologies open languages are an important aspect of implementing web services, more so than standards. It could be argued that an open-source approach and agile development methodologies can be included in this assertion.

Microsoft formed its .NET framework in the summer of 2001 [13, 56], marking its entry in to web services. .NET, or dot-net (is this a re-branding of dotcom) is essentially an upgrading of its existing suite of development technologies such as visual basic and active server pages (ASP). Still built upon the Component Object

Model (COM) .Net objectives are to build integrated distributed applications, mainly using Web technologies such as XML, an endorsement of its value. Microsoft in [13] admits the need for standards based computing architectures and it's potential for creating the semantic web. The aim of the .NET framework is to remove much of the complexity of building distributed applications in creating a common runtime environment for a range of programming languages, thus extending choice to the consumer in this case developers, who will essentially not have to learn new skills in order to use .NET. According to [56], the authors question the ability of .NET to provide sufficiently web-centric solutions. Citing as their reasoning for their technology choices for use in commercial data management systems, the authors criticise Microsoft as being dependant on the legacy of their products and the Windows platform. This results in fixed data schema, and data management based on relational models preventing innovation and flexibility. Also security issues are left unresolved and interoperability is further complicated. The approach taken by [56] involves massively simpler code, according to the authors, having eliminated much of the code need to deal with the fixed schema and predefined definitions involved with .NET approach.

In [13] the author suggests that .NET provides choices in development tools, and compares Sun Microsystems's Java as being "akin to a mechanic using a hammer to fix all repairs on your car". Others [1,11,43,44,45,47,48,77] will argue that Java is not a blunt instrument but a versatile combo tool, that has been designed from the outset to be platform agnostic and to build distributed applications. In-conjunction with scripting and plaintext mark-up languages, Java is seen as good choice for developing web based applications. It is has been used to create many of the open source software available for use in the Web. Open source and the financial cost of ownership is another aspect of the technology choices involved. Inherently free scripting languages, open source and freely available technologies such as those promoted through the Apache Foundation and others such as mySQL, and of course Java, provides a free or inexpensive alternative to .NET.

Java is easy on the programmer according to [11]. This is because of it pure object orientation along with its robust type checking and error handling. This strong typing according to [16] helps in memory allocation. Also it is not limit by the same backward compatibility issues which affect C++, being a subset of the C

programming language. Object orientation has many benefits such as reusable and interchangeable components, reducing complexity. Large but flexible libraries of code become available to the programmer. Java also has portability between platforms. This means that the programmer needs only write the code once, with no need to edit code for different platforms. This platform independence is achieved by compiling the code in a byte-code form. This is a higher level than the machine code software is typically complied with, although this is achieved at the cost of speed. Using byte-code is [16] a *halfway house* between the object orientated program and machine instructions. This means that Java has the ability the create stand alone applications or portable applets, mini programs that can be download to run with in the client browser.

What needs to be considered are the options for both server-side and client-processing, and the advantages and disadvantages of both approaches? Data aware applications are generally found on the server-side, can be used to deliver dynamic content to client whilst client-side process can be used for the presentation of download data and the validation at source of data to be uploaded.

Common Gateway Interface (CGI) is one approach to server side processing, here an executable program, stored on the server, is invoked from a web page, by following a link or clicking a submit button for example. The implementation of CGI is hidden from the user and can be written in any programming language. Another approach to server-side processing is that taken by server pages such as Microsoft's Active Server Pages and Java's JSP, as well as other technologies including Perl and PHP. Here a plaintext script is interpreted by the server which then generates a HTML page on the user's browser.

Client-side processing has the advantage of reducing the load on a server, and the dependence on any server technology, giving greater platform independence and less need for central administration. Client-side processing can take the form of Java Applets, Browser plug-in's and browser scripting languages, such JavaScript which can be embedded into the HTML of a web pages to enable programming functions to be added alongside the presentation of the page, such as validating data inputs and dynamically changing the presentation of content.

Peer-to-peer distributed application architectures have in the past suffered from difficulties of interoperability between different platforms or even different versions of the same platform as well as difficulties in information interchange between systems under different management and development teams. However where this is not the case, there are many options available to the developer looking to build a distributed system. Technologies such as CORBA and DCOM address the difficulties of interoperability. According [85] these are tightly coupled approaches, unlike web services, using XML as a message medium, which can termed as loosely coupled, and remove many of the nuances involved in creating distributed architectures. SOAP is example of XML based messaging protocol.

XML (eXtensible Mark-up language) is a plain text mark up language; this means any one can read an XML document. Other formats often require proprietary applications in order to be read, or if you are able to view the data, you may not be able to understand the formatting and structure. A machine will similarly need to be told how to interpret the format. With XML the structure and formatting is standardized and constructed in such a way that it can be read by human or machine. Another advantage of XML is the options for encoding the text, using Unicode, a 16 bit format, as a default, rather the usual 8 bit ASCII seen in other plain text formats, means a larger number of characters are available allowing for internationalization. Data can be encapsulated within the hierarchical structure of an XML document which enables complex data modelling and extensible management.

Apache foundation projects provide open source Java based and XML applications, such as Apache server, Jakarta Tomcat Server for JSPs and Xindice a native XML data server. These can be downloaded freely off the internet can be used to build the kind of XML based applications used in web services. Java's JDBC (Java Database Connectivity) package can be used to provide platform independent and remote access to distributed relational databases, providing legacy support. There are also a range of XML parsers available for use with Java. It is possible to construct a fully functioning web service architecture using essentially two skills sets, Java and XML with a third SQL to provide legacy support, an optional fourth being HTML. The Use of this software technology is essentially available free and platform independent.

2.6 Semantic Web Building

The capacity of XML to develop machine interpretable vocabularies is the essence of the Semantic Web. By including resource metadata with hyperlinks the current World Wide Web is extended beyond a publishing metaphor into a global knowledge base.

The challenge in shared knowledge management and discovery is to achieve trust and consensus in the collective view of the data and information constituting the knowledge. That is to achieve a single moment of truth at real time or near real time for data and information that is being collected and altered in dynamic and dispersed systems in which no overall central administration and dependable control is either achievable or desirable. Concurrency with in distributed computing systems can be approached with replication and version control techniques [43].

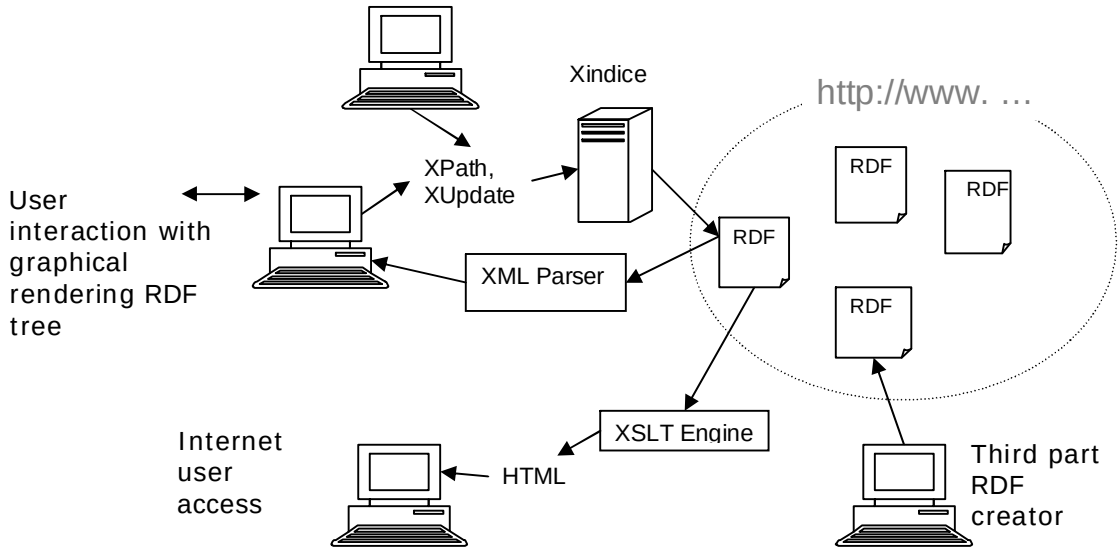
Mechanisms for analysing and cataloguing metadata, that is auxiliary information and data, (data about data), can be used to build and maintain a collective and reproducible single moment of truth, and aid in the interpolation of interrelationships between data and information to discover schema and indeed the underlying knowledge being encapsulated. This will involve the use of heuristic algorithms to crawl through the information space. To share those discoveries and make use of them, the user needs an interface through which the knowledge can be represented and enriched through interaction and manipulation. Discoveries of internal representations of conceptual data models could be presented to the user graphically and/or in a textual form, and subsequently interactions captured, through the interface, and adjustments made to the underling persistent data. In both heuristic data mining and the following graphical representation of the discovered schema, applications of Graph Theory can be found. Expedient searching of directory structure and content data and drawing of graphs are the particular challenges.

2.6.1 Global Data base Architecture

The theoretical foundations have been laid. The challenge is now to build the global database, as the semantic web is described. The use of RDF as a medium for a semantic web functioning as a global database has been discussed. The capacity of the XML/RDF combination to develop machine interpretable vocabularies is the essence of the Semantic Web. By including resource metadata with hyperlinks the current World Wide Web is extended beyond a publishing metaphor into global

knowledge base. The further combination of RDF with the Dublin Core vocabulary will add further standards for describing and discovering metadata. Any system for building and managing RDF documents can be developed as an open system and may benefit from being operated as a web service, as the authors of the RDF documents will probably wish to share and distribute their documents, and as has been discussed, the web service approach has many benefits. Such approaches are not mandatory. RDF documents can be created and managed in other whys.

The following diagram suggests an implementation for building an interconnected collection RDF documents, a portion of the semantic web. Here a Native XML Database is the document management system, giving concurrency and semi-permanence to documents shared by a community of users.



The Global databases will consist of a multitude of RDF documents independently created by many authors using different systems and approaches. The nature of these documents means that they can be parsed by anyone with the technology and understanding of the standards. Using an XML parser the user can browse the document without a prori of the schema or vocabulary. It therefore follows that client applications of our document management system are not schema dependent. The underlying data structure is not only standardised it is also extensible. Authors of documents in our document management system can, likewise, parse then browse

and edit documents without knowledge of and explicit reference to a fixed schema. In conventional data management systems, as have been discussed, developers and administrators need to know the details of the schema.

In order to edit and update these discovered documents there needs to be a means to explore their contents. This could entail graphical rendering of structure and content that can be interacted with. The challenges and approaches to the graph drawing have been considered, [32, 76] along with applications in representing information and data. A simple and commonly used mechanism is to represent hierarchal data in a tree form, as use for exploring directory structures. Here nodes can be explodes or collapsed to reveal lower level branch nodes, other directories and leaf nodes, files. JTree is the Java Swing component, it is highly customisable. It can be populated with hierarchal data which is then displayed vertically with the nodes in rows. Here the elements and data parsed from the document can sequentially populate the tree. Mouse actions and other events can be used to capture the user's intentions and the relevant XPath statement, creating in turn, instruction to be passed back to the XML database, in the case Xindice, an Xupdate statement. The XML database handles transactional, access and concurrency issues, the front-end client will handle refreshing views.

Security issues are tried and tested. Sensitive document collections can be stored on a secure server, using for example SSLs. The validity of integrity of individual documents can be verified using digital signatures. The Dublin Core elements can be used in the documentation of the system aiding version control and asserting intellectual property rights.

A remaining issue is that of the validity of links, internal and external. Referential integrity can not be strictly enforced in the case of external links. Here a weak updating mechanism will be necessary. This could involve a software agent, similar to search engine robots, discovering changes. This update demon could be a background service initialled by users or running on a separate server.

The generation of an RDF documents may be by machine as well as by human authorship. Dynamic information elicitation by intelligent agents, crawling the web,

or data exported from legacy RDBMS systems and formatted in RDF. The construction of the semantic web will necessarily be decentralised and distributed.

2.7 Conclusions and Future work

2.7.1 Future Work

The aims of this work were to explore and document the means to create collaborative and extensible systems for the management of distributed data and metadata, building a semantic layer for distributed knowledge management. Informatics is the science of processing and managing data and information, then we could perhaps defined define the processing and management of distributed data, metadata, information and knowledge representation as *teleinformatics*. Thus this work has been an exploration of the tools, techniques and practise of teleinformatics.

Given that XML is the main ingredient, and a natural format for databases, much focus has been on the means to use and manage XML data. When this project was first considered, and given this fast moving area of technology, there did not appear many solutions to the management of XML data. XML was not much more than a good idea waiting to be used, this may no longer be strictly true, with .NET coming on to the market , it is still however a new technology awaiting applications. In the case of RDF and Semantic Web, there is like-wise a good idea waiting for applications.

The contribution of this work is hopefully to provide a guide to those seeking to prove the admittedly broad and sweeping concepts discussed. Requirements, technologies and some of the fundamentals have been discussed alongside a business case. The economic context in which this work has been framed is one of destructuring, decentralizing organisations becoming agile and adaptive to increasingly complex market conditions, with rapid technological change. Should we be testing this assumption? The alternative is increasing structure and bureaucracy, centralisation and control, either way the justifications for adoption of the approaches discussed as the same.

Prove of concept, in this case, are applications and web services contributing to the semantic web, in effect creating the global database. These could include:

- Agents, to scan information space, to catalogue metadata, dynamically building RDF documents, and verifying the validity of links.
- Software for analysing and mapping legacy data and schema, discovery and documenting relationships.
- Interactive RDF management systems, enabling human authorship of new RDF documents, and the browsing and editing of exiting documents.
- And Web Services for managing the discovery, sharing and distributing RDF Documents.

Anyone using this work, as a guide in creating any aspect of the above, will need to start of by looking for the latest state of play, in technology areas discussed some progress in this field is inevitable, and ideas contained herein will be subject to obsolescence but we hope are a worth while exercise, even if only from a historical perspective.

2.7.2 Conclusions

Standardisation is an indicator of progress, weights and measures, time, railway gauges, electrical plugs and sockets, accountancy, cargo containers, QWERTY keyboards, recoding media such as LPs, VHS cassettes, CD-ROM, DVD's etc are just some examples. Some people may not like standardisation, but it does usually, in the long run, prove to be for the greater good. It allows for people portability and the interoperability of technology giving greater overall adoption of the underlying systems involved. Standard railway gauges, for instances [105], lead to the creation of interconnect railway networks, as did standards for internetworking lead to the creation of the internet and World Wide Web.

The IT industry as we have argued, knows this, yet there is a proliferation of innovations vying for dominance. No one wants ends up the owner of redundant technology, such as in the case of railways, with the wrong size of track. These innovations be they hardware gadgets, software applications, programming languages, methodologies, protocols or XML vocabularies all seek standardisation bringing wealth and status to their inventors, as Microsoft enjoys. It could be argued that pursuit of fortune and glory spurs on these innovators and entrepreneurs. This may be true, but an alternative view point would suggest the locking away of blue

prints and the slapping on of patents, which can for some products, hamper market acceptance.

At the end of the 20th century, IT was seen as a pot of gold. With vast profits for the entrepreneurs and their investors, and well paid employment for those with the skills. At the turn of century, the so called bubble proverbially burst. There are many reasons, whys and where fors of this, beyond the scope of this work. Many of the apparent promises of riches lay in the internet. However the creation of internet stems from its not-for-profit nature and its success in its accessibility, mass market adoption and its powers to save its users money, through the reduction of transaction cost and symmetry of information. In any proverbial gold rush, those making money are those selling the proverbially shovels (Internet Explorer, Google etc.).

In the case of knowledge, knowledge not shared is not knowledge at all. Data, Information need not be exposed to whole world, it may be shared within a secure corporate environment, even in this case there are benefits in having a standard medium for storing and exchanging knowledge. RDF is one such medium. Open standards and system as we have seen, provide interoperability of hardware, software and peopleware. People become more portable; skills more easily acquired and transferred within one organisation can be taken into other. With the technology, less time is needed to configure and integrate. Extensibility and schema independences also free man hours, as do web based platform independent solutions. Decentralised infrastructures are more robust. And RDF/XML vocabularies can be used by machines creating greater automation. Streamlining IT Departments? Bureaucracies? Or saving users from technical glitches and administrative chores and empowering then to think and create in ways a machine cannot yet do.

Bibliography

References

[1] Silis, A. Hawick, K. "World Wide Web Server Technology and Interfaces for Distributed, High-Performance Computing Systems" August 1997, DISCWorld DHPC017

[2] Hawick, K. James, H. Vaughan, F. Patten, C. "DISCWorld: A Distributed High Performance Computing Environment" November 1997, DISCWorld DHPC020

[3] Cordy, O. Dieng, R. Hebert, C. "A Conceptual Graph Model for W3C Resource Description Framework" 1999, http://www.inria.fr/acacia

[4] Rabarijaona, A. Dieng, R. Cordy, O. "Building a XML-based Coperate Memory" 1999, http://www.inria.fr/acacia

[5] Hawick, K. James, H. "Querying and Auxiliary Data in the DISCWorld" January 1998, DISCWorld DHPC030

[6] Hawick, K. James, H. "Modeling A Gossip Protocol for Resource Discovery in Distrbuted Systems" 2000, DISCWorld DHPC102

[7] Coddington, P. James, H. Hawick, K. "An Environment for Workflow Applications on Wide-Area Distributed Systems" June 2000, DISCWorld DHPC91

[8] Patten, C. Vaughan, F. Brown, A. Hawick, K. "DWorFS: File System Support for Legacy Applications in DISCWorld" January 1998, DISCWorld DHPC32

[9] James, H. Hawick, K. "Data Futures in DISCWorld" 2000, DISCWorld DHPC083

[10] Johannesson, P. "Using Conceptual Graph Theory to Support Schema Integration" 2000,

[11] Eckel, B. "Thinking in JAVA 2nd edition" 2000, Prentice Hall

[12] "Web Services, Has IT found Nirvana" May 2002, www.cw360.com

[13] Greatorea, M. "Microsoft: We're open for webservices" December 2001, www.cw360.com

[14] Peter, J. "The Importance of being open" December 2001, www.cw360.com

[15] Papazoglou, M. Proper, H. Yang, I. "Landscaping the Information Space of large Multi-Database Networks" Feburary 2001, Data and Knowledge Engineering 36(3):251-281

[16] Wright, C. "Teach Yourself Java" 1997, Hodder and Stoughton

[17] Wu, M. Lee, C. "On Concurrency Control in Multidatabase Systems" 1998, National Cheng-Kung University

[18] Eddy, S. DeLong, B. "XML in plain English, 2nd Ed" 2001, M and T Books

[19] Carlson, D. "Modeling XML Applications with UML" 2001, Addison Wesley

[20] Oestreich, B. "Developing Software with UML, Object Oriented Analysis and Design in Practice" 1999, Addison Wesley

[21] Franklin, S. "Artifical Minds" 1997, MIT Press

[22] Jannink, J. Mitra, P. Neuhold, E. Srinivasan, P. Studer, R. Wiederhold, G. "An Algebra for Sematic Interopation of Semistructured Data" 1999, Standford

[23] Cohen, W. "Intergration of Heterogenous database without common domains using quires based on textual similarities" 1998, AT and T

[24] Genesereth, M. "Knowledge Interchange Format" 1998, draft proposed American National Standard (dpANS) NCITS.T2/98-004

[25] Ananthaswamy, A. "You Hum it and i'll fund it" 17th March 2001, New Scientist vol169 no 2282

[26] Toonen, P. Bommel, P. "Markov Chain Fundamental for Data Schema Transformations" May 1995, CSI-R9505

[27] Bommel, P. Lucasius, C. van der Weide, T. "Pool Heuristics in Evolutionary Database Optimization" October 1994, CISMOD 94

[28] Patten, C. Hawick, K. "Flexible High-Performance Access to Distributed Storage Resources" 2000, DISCWorld DHPC-085

[29] Asai, T. Abe, K. Kawasasoe, S. Arimura, H. Sakamoto, H. Arikawa, S. "Efficient Substructure Discorvery from Large Semi-structured Data" October 2001, SDM2002

[30] Turau, V. "Making Legacy data accessible for XML applications" 2000, DB2XML

[31] Bowers, D. "From Data to Database" 1988,

[32] Cohen, R. Battista, G. Tamassia, R. Tollis, I. "A Framework for Dynamic Graph Drawning" August 1992, CS-92-34

[33] Cover, R. "The XML Cover Pages" February 2002, http://xml.coverpages.org/

[34] Odasanjo, D. "An Exploration of XML in Database Management Systems" 2001,

[35] "IP Storage: The New Age Frontier" June 2001, computerworld white paper Computerworld

[36] Butler, M. "The Way ahead with with Web services" January 2002, cw360.com

[37] Becket, H. "Making XML a shared Experience" January 2002, cw360.com

[38] Bradbury, D. "ebXML: setting the standard" November 2001, cw360.com

[39] "Transparent Persistence" 2002, Sun Microsystems inc. Sun One Studio

[40] Megginson, D. "SAX Documentation" July 2001,

[41] "Web Services Made Easier" October 2001, Sun Microsystems, inc.

[42] "Data Access Objects" 2001, Sun Microsystems, inc.

[43] Coulouris, G. Dollimore, J. Kindberg, T. "Distributed Systems, Concepts and Designs" 2001, Addison Wesley

[44] Lea, D. "Concurrent Programming in Java, 2nd Ed" 1999, Sun Microsystems inc.

[45] Hardy, V. "Java 2D API" 2000, Sun Microsystems inc.

[46] Kling, A. "Economics and the Transparent Society" 2000, "Arguing in my spare time" No.4.32 http://arnoldkling.com

[47] Budd, T. "Classic Data Structures in Java" 2001, Pearson Education

[48] Weiss, M. "Data Structures and Algorithm Analysis in Java" 1999, Addison Wesley

[49] Cox, D. Smith, W. "Queues" , Monograph on applied prod. Chapman and hall

[50] Wilson, R. "Introduction to Graph Theory" 1996,

[51] "The Need for a shared Ontology" 2000, Ontology.org

[52] Sanders Peirce, C. "Existential Graphs" 2002, Commentry by John F. Sowa MS 514

[53] Mancini, J. "Having problems controlling your enterprise documents? ECM can Help" 2001, AIIM International Computerworld

[54] "Enabling the Digital Ecosystem" 2001, Silicom.com http://www.mediaswitch.com/A5561D/Home.nsf/DE%20In

[55] May, T. "Sailing the Seven C's of Knowledge Management" 1996, Computerworld

[56] "CQR Data vs .Net" 2001, www.cqrdata.com CQR Data

[57] Loshin, P. "Knowledge Management" October 2001, computerworld

[58] Vivek, G. "An Introduction to Data Warehousing" 2001, Systems Services Cooperation http://www.system-services.com/dwintro.asp

[59] Berners-Lee, T. "Web Architecture from 50,000 feet" 2002, w3c.org www.w3.org/DesignIssues/Architecture.html

[60] Bourret, R. "XML and Databases" 2001,

[61] "Applications for Cyc" March 2002, Cycorp

[62] Curly, K. "10 Myths About Knowledge Management" Janaury 2001, Computerworld

[63] "Emergence News european survey" 2001, www.emergence.nu

[64] Ambsioa, J. "Knowledge Management Mistakes" July 2000, Computerworld

[65] "CORBA and XML; conflict or cooperation?" 2001, www.omg.org Object Management Group

[66] "CORBA Basics" 2001, www.omg.org Object Management Group

[67] "Web mining" October 2000, IT-director.com

[68] "CRM and Data" May 2001, IT-director.com

[69] "Professional XML" 1998, Wrox Press from www.XMLdude.com chapter 6

[70] Roman, S. "Access Database, Design and Programming" 1997, O Reilly

[71] Davis, G. Naumann, J. "Personal Productivity with Information Technology" 1997, McGraw-Hill Book Co

[72] Goodrich, M. Kobourov, S. "VLGD, Algorithm for Drawning Very Large Graphs" August 1997,

[73] Lyons, K. Meijer, H. Rappaport, D. "Algorithms for Cluster Busting in Anchored Graph Drawing" 1997, Vol 2. no. 1 pp.1-24

[74] Brandes, U. Kaab, V. Loh, A. Wagner, D. Willhalm, T. "Dynamic WWW Structures in 3D" 2000, Vol 4. no. 3, 183-191

[75] Sengupta, A. "Design and Implementation of a Database Environment for the Manipulation of Structured Documents" May 1995,

[76] Ceaser, . "Graph theorectical foundation classes and a framework for graph drawing" 1999, www.aplhaworks.ibm.com/tech/gfc

[77] Rockwell, W. "XML, XSLT, Java, and JSP" 2001, New Raiders

[78] Bigus, J. Bigus, J. "Constructing Intellegent Agents Using JAVA, 2nd ED" 2001, Wiley

[79] Ayres, R. "Professional Issues in Computing" 1999, Prentice hall

[80] Parkes, S. "Cooking the search books" November 2001, FTIT (Financial Times)

[81] Bruemmer, P. "Search Engine Marketing: The Outsource Advantage" 2001, Web Ignite

[82] Fifield, C. "Effective Search Engine Design" November 2002, SearchDay, Number 394

[83] Rappoport, A. "Anatomy of a Search Engine" October 2002, SearchDay Number 388

[84] Ashworth, C. Goodland, M. "SSADM a practical approach" 1990, McGraw-Hill

[85] "What are Web Services" May 2002, IT-director.com

[86] Fowler, M. "The New Methodology" June 2002, www.martinfowler.com

[87] Sengupta, A. "Design and Implementation of a Database Environment for the Manipulation of Structured Documents" May 1995,

[88] Lawton, G. "The Great Giveaway" Feburary 2002, New Scientist Vol 173 No 2328

[89] "Prince 2, Overview" November 2002, www.ogc.gov www.ogc.gov/prince

[90] Wells, D. "Extreme Programming" December 2002, www.extremeprogramming.org www.extremeprogramming.org

[91] "Dynamic Systems Development Methodolgy" December 2002, www.dsdm.org www.dsdm.org/en/about.asp

[92] "The ITIL, ITSM & Security Directory" 2002, www.itil-itsm-world.com

[93] Loy, M. Eckstein, R. Wood, D. Cole, B. "Java Swing, 2nd Edition" 2003, O Reilly

[94] Reese, G. "Database Programming with JDBC and Java" 2000, O Reilly

[95] Brown, R. "Understand Industrial Organisations" 1992, Routledge

[96] Gritfiths, A. Wall, S. "Applied Economics, Sixith Edition" 1995, Longman

[97] "Network Effect/Reed's Law" March 2003, Wikipedia.org

[98] Byod, C. "Metcalf's Law" March 2003, MGT 487 On Line

[99] "Metcalf's Law" 1999, SearchNetwork.com

[100] Downes, L. Mui, C. "Unleashing the Killer App" 2003, www.killer-apps.com

[101] Gilder, G. "Metcalfe's Law and Legacy" 1993, Telecosm Series

[102] Watson, R. "Data Management, Database and Organizations" 1999,

[103] Albert, J. "Theorectical Foundation of Schema Restructuring in Heterogeneous Multidatabase systems" 2000, ACM

[104] Raymond, E. "The Cathedral and Bazaar" 1998, Published in First Monday

[105] Funk, K. Jordon, J. "The Quite Internet Revolution: Beyond Transplanted Transactions to Networked Innovations" 2000, Center for Business Innovations

[106] Lewin, R. Parker, T. Regine, B. "Complexity Theory and the Organization: Beyond the Metaphor" 2002, Center for Business Innovations

[107] "Resource Description Framework (RDF) Model and Syntax Specification" Feburary 1999, w3c http://www.w3.org/TR/1999/REC-rdf-syntax-19990222

[108] Gray, P. Hui, K. Preece, A. "An Expressive Constraint Language for Semantic Web Applications" 2002, University of Aberdeen

[109] "Product Overview for .NET Framework 1.1" 2003, Microsoft msdn.microsoft.com

[110] Tapang, C. "Web Services Description Langauge (WSDL) Explained" July 2001, Microsoft msdn.microsoft.com

[111] "UDDI" May 2003, www.uddi.org

[112] "The Economics of Information" April 2003, www.serverworldmagazine.com/contra/2000/2q/06_th

[113] "Thriving in an Uncertain Economy" September 2001, Point of View, Cap Gemini Enrest & Young

[114] "Markets with Asymmetric Information" Octoder 2001, Nodel_paper.pdf

[115] Berners-Lee, T. "Semantic Web Road map" September 1998, w3c www.w3.org/DesignIssues/Semantic.html

[116] Berners-Lee, T. "Metadata Architectecture" January 1997/Sept 2000, w3c www.w3.org/DesignIssues/Metadata.html

[117] Berners-Lee, T. "Relational Databases on the Semantic Web" Sept 1998/March 2002, w3c www.w3.org/DesignIssues/RDB-RDF.html

[118] Berners-Lee, T. Hendler, J. Lassila, O. "The Semantic Web" May 17, 2001, Scientific American

[119] "Dublin Core Metadata Initiative Overview" Feburary 2003, http://dublincore.org/about/

[120] Bray, T. "What is RDF?" 2001, www.xml.com/pub/a/2001/01/24/rdf.html

[121] MaKenna, E. "Business Psychology & Organisational Behaviour" 1994, Lawrence Erlbaum Associates Ltd

[122] Williams, S. Perry, W. "White Paper: What is an XML Database?" 2001, CQR data

[123] Bourret, R. "XML Database Products" August 2003,

[124] Pascal, F. "Managing data with XML: Forward to the past?" January 2001, http://searchdatabase.techtarget.com/tip/1,289483, FABIAN PASCAL'S AGAINST THE GRAIN

[125] Pascal, F. "XML data management: Setting some matters straight" June 2001, http://searchdatabase.techtarget.com/tip/1,289483, FABIAN PASCAL'S AGAINST THE GRAIN

[126] Kephart, J. Chess, D. White, S. "Computers and Epidemiology" May 1993, IEEE SPECTRUM

[127] Chess, D. "The Future of Viruses on the Internet" Octoder 1997, Virus Bulletin International Conference

[128] Carr, K. "Sophos Anti-Virus detection: a technical overview" Octoder 2002, www.sophos.com

[129] Sengupta, A. "Towards the Union of Databases and Document Management" 1998, DocBase

[130] "Xindice Documentation" March 2002, Apache Software Foundation xml.apache.org/xindice

[131] Staken, K. "Xindice Developers Guide" March 2002, www.dbxml.org/docs

[132] "Xerces Java Parser Readme" May 2001, Apache Software Foundation Xerces 1.4.0

[133] "XML:DB Initiative: Enterprise Technologies for XML Databases" March 2002, XML:DB www.xmldb.org

[134] Martin, L. "Requirements for XML Update Langauge" November 2000, XML:DB www.xmldb.org/xupdate

[135] Andreas, L. Martin, L. "XUpdate, XML Update Langauge" September 2000, XML:DB www.xmldb.org/xupdate

[136] Staken, K. "XML Database API Draft proposal" September 2001, XML:DB www.xmldb.org/xapi

Index