Systems Development in a GRIDs Environment

Keith G Jeffery
Director IT and International Strategy
CCLRC Rutherford Appleton Laboratory
Chilton, Didcot, OXON OX11 0QX UK
k.g.jeffery@rl.ac.uk

Abstract
Over the past 30 years or more information systems engineers have attempted to improve the cost effectiveness of systems development by improving requirements capture and analysis, by structured design, by utilising design languages that can be verified for consistency more or less formally and in some cases matched formally to requirements. While data design methods improved significantly with relational and extended relational paradigms, program design was not so successful. Jackson input-process-output and hierarchic design methods gave way to functional. Object-orientation soon came up against the inability of hierarchic / inheritance mechanisms to represent the real world requirements which has more complexity. Aspect-oriented programming was intended to resolve this problem but appears to have caused even more confusion. Meantime, a bringing together of functional and object-oriented process design as service-oriented architecture, together with relational data design principles, has given some hope for progress.

Early system design achieved device independence of programs and then (with relational technology) true data independence. However, general virtualisation of computing, data storage and communications resources has hitherto not been possible. The GRIDs paradigm achieves this latest step forward. Starting with metacomputing (linked supercomputers) in the USA, the European vision of GRIDs is a general IT 'surface' with which the end-user interacts intelligently to determine her requirement and the system behind the surface offers a 'deal' to fulfil the request. Beneath the 'surface' various architectures have been attempted.

The GLOBUS architecture provides computational scheduling, but does not virtualise generally computation, data or network resources. The bringing together of WS (web services) with the GRIDs environment led to OGSA (Open Grids Services Architecture). Work with OGSA has exposed two major problems: the operating system facilities provided today are inadequate in various areas including security and resilience and the multiple layers in the service-oriented architecture expose too much complexity. The latest thinking revolves around SOKU (Service Oriented Knowledge Utilities) which are composed of self-managing, self-assembling, self-organising and self-destroying processes with exposed parametric and data input/output interfaces as well as its service description including non-functional aspects. The key is metadata (describing the SOKU processes and the data resources) and its use.

1 A SHORT HISTORY OF COMPUTING: A PERSONAL VIEW

From the earliest stages of using computers there has been the concept (although not the name) of systems engineering. Even the earliest programmers planned out their program before coding it. The software systems development methods developed along two parallel lines: one emphasising efficiency of developer time and fidelity to informally-defined user requirements with associated techniques of prototyping and fourth generation languages; the other emphasising correctness of the program, formal verification and formally-defined specifications. The former led to the so-called design methodologies, the latter to formal methods in software engineering, proof systems and in particular their application in the safety-critical environment.

An alternative and complementary viewpoint emerged in the sixties where systems development concentrated on the data. This was no surprise since large corporations were using computers for business processing and required to represent their business world of interest.

The problems with these approaches used in the seventies and eighties are well-documented. Software development was slow and error-prone and the use of formal methods made it slower. Data-centric approaches failed to map correctly the objects and their relationships in the real world: well-known examples include early database systems which could only map hierarchies, not fully-connected graph structures.

With the relational theory of data, and attached concept of the relational algebra (and calculus) a new age dawned. By the late eighties the first kind of software engineering came together with data engineering providing unprecedented speeds of system development, conformance to informally-stated user requirements and ability to adapt. Security issues emerged and were solved and there were attempts at distribution and the provision of business continuity. Furthermore, entity-relationship modelling based on the relational approach provided a further level of abstraction and a communications environment between the designer/developer and the end-user. Earlier work on artificial intelligence became encapsulated as knowledge engineering and aided the modelling process by providing a formalism for expressing the semantics of the information and for specifying constraints. In this era Arne Sølvberg and his team produced excellent R&D results demonstrating formal systems engineering from requirements specification to running system.

However, early systems had problems with performance, and errors made in the data modelling led to many work-arounds and modifications to the associated software. The systems became expensive to maintain. Steps were taken to formalise requirements and to generate systems – both data structures and software – using predefined component software fragments. The concept now known as services emerged.

To overcome the data-process gap, object-orientation was introduced reaching acceptance in the late eighties. Based on much earlier software engineering principles (eg those of Simula in the sixties) object-orientation encapsulated the static data model aspects of the application with the dynamic process aspects. Information Systems developed using this paradigm proved to be lacking in performance and additionally there were problems in both data modelling (related to hierarchic restrictions on inheritance) and process-modelling (repetition of the same code for many objects). The latter was to some extent overcome by aspect-oriented programming at the end of the nineties but by then the world was returning (indeed, many had never left) to relational database technology (improved with some object-oriented aspects).

The emergence of the world-wide-web rekindled interest in old technologies in information retrieval and hypermedia. Progressively the web systems developed and heightened the visibility of technologies such as mobile code, service-oriented architecture and hypermedia. The web offered new possibilities in user interface design and in thin clients. Linking with the emergence of wireless technology and widespread use of mobile phones it was a short step to the concept of ambient, pervasive computing. Tim Berners-Lee emphasised the importance of semantics and trust rekindling interest in knowledge engineering and formalised systems respectively.

In the USA in the late nineties the need for massive computation power to simulate various physical phenomena (from nuclear explosions to climate) led to the metacomputing (linked supercomputers) concept popularised as 'the GRID'. In Europe a wider concept emerged simultaneously – GRIDs. It is in this context that systems engineering – and specifically information systems engineering - is now discussed.

2 REQUIREMENTS TODAY AND TOMORROW

2.1 User Perspective

Users demand systems that are easy to use. The end-user has a low attention span and requires immediate satisfaction. The threshold barrier to achieve usage of the system must be very low, else impatience precludes usage. The system should be capable of handling heterogeneous character sets, languages (information syntax) and semantics (knowledge). The system must be easy to understand and intuitive in its operation. The system must be knowledge-assisted – providing contextual hints, help, explanation. It must also be knowledge-assisted to assist in reducing the effort of input and update and to ensure constraints are in place to assure data quality. The end-user must have choice in the end-user device used implying that the system must be device-independent, adaptive to changing user modes of interaction through various media and be adaptable for variously-abled persons. Naturally the system must handle the problems of intermittent connection, synchronisation and data transfer optimisation required in a mobile, ambient, pervasive environment. This requires particular expertise in user interface design.

2.2 System Perspective

The system should provide adequate performance, achieved by (re-)scheduling, parallelism, and distribution with appropriate optimisations and reconfiguration. The system should provide appropriate security, trust, privacy. For security of future use, the system should provide appropriate curation, preservation, which may be linked with appropriate failover and business continuity facilities. All this implies that systems have to be 'self-*' or autonomic: self-managing, self-configuring, self-organising, self-tuning, self-scheduling, self-maintaining, self-adapting. With millions of nodes and massive processing requirements it will be simply uneconomic for persons to 'be in the loop' of systems management.

3 THE GRIDS PARADIGM

3.1 A brief history

The concept of the GRID was initiated in the USA in the late 1990s. Its prime purpose was to couple supercomputers in order to provide greater computational power and to utilise otherwise wasted central processor cycles. Starting with computer-specialised closed systems that could not interoperate, the second generation consists essentially of middleware which schedules a computational task as batch jobs across multiple computers. However, the end-user interface is procedural rather than fully declarative and the aspects of resource discovery, data interfacing and process-process interconnection (as in workflow for a business process) are primitive compared with work on information systems engineering involving, for example, databases and web services.

Through GGF (Global GRID Forum, now OGF Open GRID Forum) a dialogue has evolved the original GRID architecture to include concepts from the web services environment. OGSA (Open Grid Services Architecture) with attendant interfaces (OGSI) is now accepted by the GRID community and OGSA/DAI (Data Access interface) provides an interface to databases at rather low level.

In parallel with this metacomputing GRID development, an initiative started in UK has developed an architecture for GRIDs that combines metacomputing (i.e. computation) with information systems. It is based on the argument that database R&D (research and development) – or more generally ISE (Information Systems Engineering) R&D - has not kept pace with the user expectations raised by WWW. Tim Berners-Lee threw down the challenge of the semantic web and the web of trust [1]. The EC (European Commission) has argued for the information society, the knowledge society and the ERA (European Research Area) – all of which are dependent on database R&D in the ISE sense. This requires an open architecture embracing both computation and information handling, with integrated detection systems using instruments and with an advanced user interface providing 'martini' (anytime, anyhow, anywhere) access to the facilities. The GRIDs concept [6] addresses this challenge, and further elaboration by a team of experts has produced the EC-sponsored document 'Next Generation GRID' [3].

It is time for the database community (in the widest sense, i.e. the information systems engineering community) to take stock of the research challenges and plan a campaign to meet them with excellent solutions, not only academically or theoretically correct but also well-engineered for end-user acceptance and use.

3.2 The Idea

In 1998-1999 the UK Research Council community was proposing future programmes for R&D. The author was asked to propose an integrating IT architecture [6]. The proposal was based on concepts including distributed computing, metacomputing, metadata, agent- and broker-based middleware, client-server migrating to three-layer and then peer-to-peer architectures and integrated knowledge-based assists. The novelty lay in the integration of various techniques into one architectural framework.

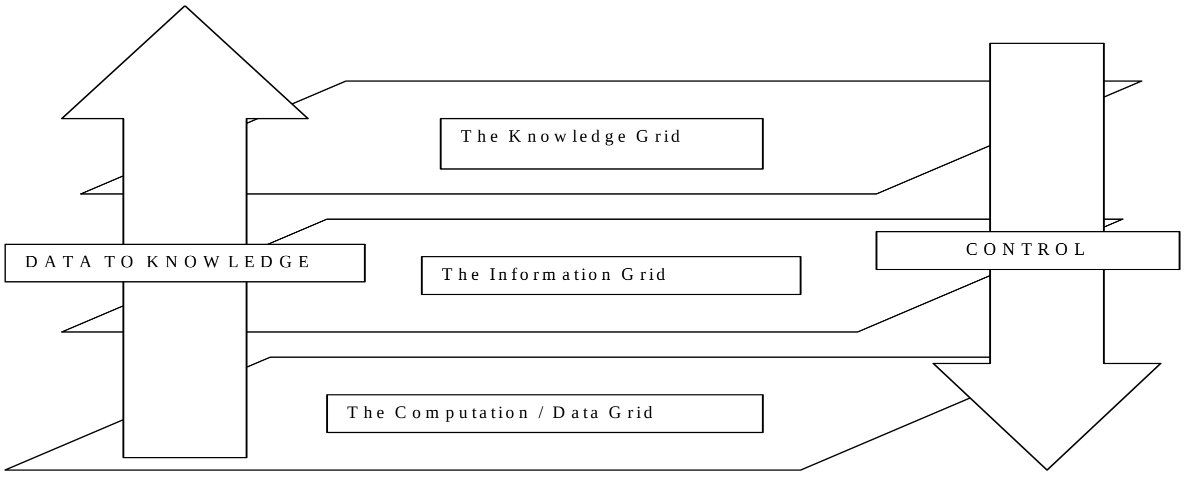

Fig. 1. Grids Architecture

3.3 The Requirement

The UK Research Council community of researchers was facing several IT-based problems. Their ambitions for scientific discovery included post-genomic discoveries, climate change understanding, oceanographic studies, environmental pollution monitoring and modelling, precise materials science, studies of combustion processes, advanced engineering, pharmaceutical design, and particle physics data handling and simulation. They needed more processor power, more data storage capacity, better analysis and visualisation – all supported by easy-to-use tools controlled through an intuitive user interface.

On the other hand, much of commercial IT (Information Technology) including process plant control, management information and decision support systems, IT-assisted business processes and their re-engineering, entertainment and media systems and diagnosis support systems all require ever-increasing computational power and expedited information access, ideally through a uniform system providing a seamless information and computation landscape to the end-user. Thus there is a large potential market for GRIDs systems.

The original proposal based the academic development of the GRIDs architecture and facilities on scientific challenging applications, then involving IT companies as the middleware stabilised to produce products which in turn could be taken up by the commercial world. During 2000 the UK e-Science programme was elaborated with funding starting in April 2001.

3.4 Architecture Overview

The architecture proposed consists of three layers (Fig.1). The computation / data grid has supercomputers, large servers, massive data storage facilities and specialised devices and facilities (e.g. for VR (Virtual Reality)) all linked by high-speed networking and forms the lowest layer. The main functions include compute load sharing / algorithm partitioning, resolution of data source addresses, security, replication and message rerouting. This layer also provides connectivity to detectors and instruments. The information grid is superimposed on the computation / data grid and resolves homogeneous access to heterogeneous information sources mainly through the use of metadata and middleware. Finally, the uppermost layer is the knowledge grid which utilises knowledge discovery in database technology to generate knowledge and also allows for representation of knowledge through scholarly works, peer-reviewed (publications) and grey literature, the latter especially hyperlinked to information and data to sustain the assertions in the knowledge.

The concept is based on the idea of a uniform landscape within the GRIDs domain, the complexity of which is masked by easy-to-use interfaces.

3.5 The GRID

In 1998 – in parallel with the initial UK thinking on GRIDs - Ian Foster and Carl Kesselman published a collection of papers in a book generally known as 'The GRID Bible' [4]. The essential idea is to connect together supercomputers to provide more power – the metacomputing technique. However, the major contribution lies in the systems and protocols for compute resource scheduling. Additionally, the designers of the GRID realised that these linked supercomputers would need fast data feeds so developed GRIDFTP. Finally, basic systems for authentication and authorisation are described. The GRID has encompassed the use of SRB (Storage Request Broker) from SDSC (San Diego Supercomputer Centre) for massive data handling. SRB has its proprietary metadata system to assist in locating relevant data resources. It also uses LDAP as its directory of resources. The GRID corresponds to the lowest grid layer (computation / data layer) of the GRIDs architecture.

4 THE GRIDS ARCHITECTURE

4.1 Introduction

The idea behind GRIDs is to provide an IT environment that interacts with the user to determine the user requirement for service and then, having obtained the user's agreement to 'the deal' satisfies that requirement across a heterogeneous environment of data stores, processing power, special facilities for display and data collection systems (including triggered automatic detection instruments) thus making the IT environment appear homogeneous to the end-user.

Referring to Fig. 2, the major components external to the GRIDs environment are:
a) users: each being a human or another system;
b) sources: data, information or software
c) resources: such as computers, sensors, detectors, visualisation or VR (virtual reality) facilities

Each of these three major components is represented continuously and actively within the GRIDs environment by:
1) metadata: which describes the external component and which is changed with changes in circumstances through events
2) an agent: which acts on behalf of the external resource representing it within the GRIDs environment.
As a simple example, the agent could be regarded as the answering service of a person's mobile phone and the metadata as the instructions given to the service such as 'divert to service when busy' and / or 'divert to service if unanswered'.

Finally there is a component which acts as a 'go between' between the agents. These are brokers which, as software components, act much in the same way as human brokers by arranging agreements and deals between agents, by acting themselves (or using other agents) to locate sources and resources, to manage data integration, to ensure authentication of external components and authorisation of rights to use by an authenticated component and to monitor the overall system.
From this it is clear that they key components are the metadata, the agents and the brokers.

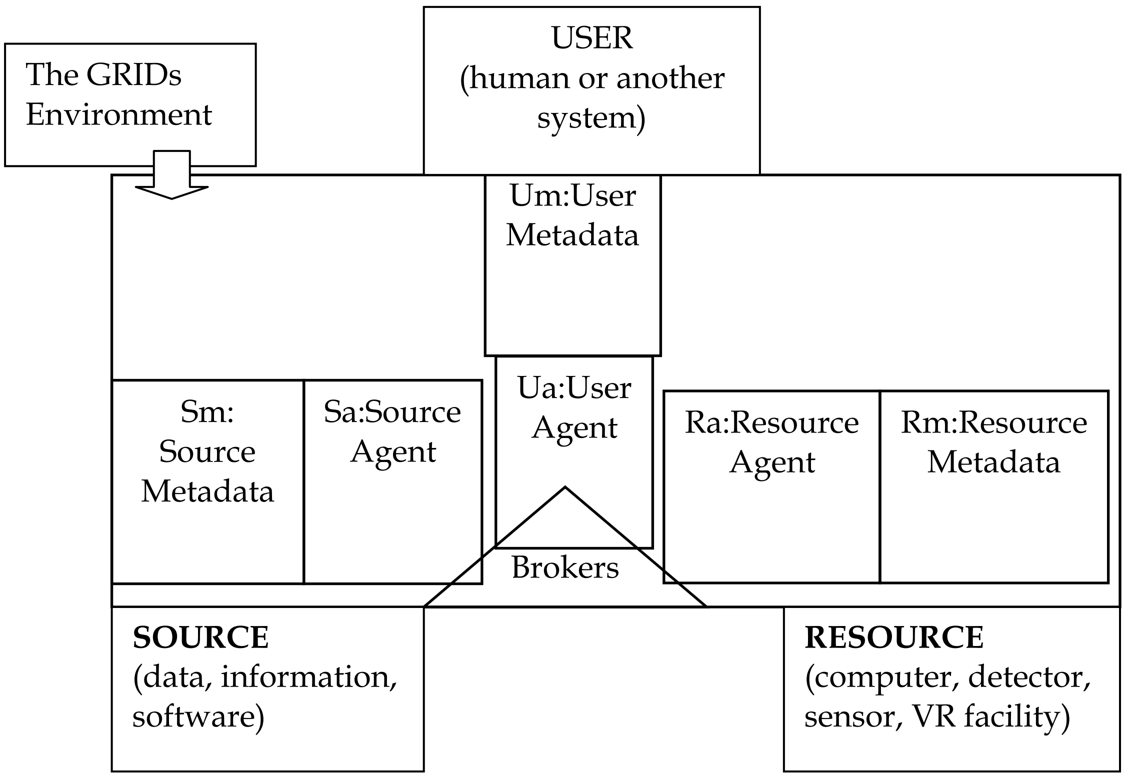

Fig. 2. The GRIDs Components

5 SYSTEMS DEVELOPMENT IN A GRIDS ENVIRONMENT

5.1 Introduction

The architecture sketched above depicts the middleware necessary for applications to be constructed and executed. The EC NGG1 (Next Generation Grids Expert Group 1) confirmed this set of requirements and architecture characterised as 'the invisible GRID' i.e. hidden from the end-user but available and performing. However the requirements outlined earlier demand 'so-called non-functional' characteristics of the system which are not at this level but which concern the lower levels of the architecture – in particular the provision of performance and trust /security. This led the NGG2 expert group of the EC to an architecture for systems development with operating systems enhanced with foundationware to bring them up to the required interface standard including provision of trust and security, performance and self-* features. Above the foundationware is the service-oriented middleware providing the basic services required by end-user applications which themselves sit on top of the middleware layer and are developed essentially by composition of services including, where necessary because of unavailability, the provision of new services.

It was then realised by the NGG3 expert group of the EC that this layered architecture was too complex and that a simplification was possible. The foundationware and middleware layers could both be implemented as components providing services, and these components had to have certain characteristics. Essentially the components had to be active, that is they had to be themselves self-motivating such that they could compose themselves into applications based on requirements, and since they are self-managing they could reorganise (self-tune, self-schedule) with changing resource availability opportunities and changing user requirements. The SOKU (service-oriented knowledge utility) concept was defined.

5.2 SOKU

The concept of SOKU is based on a service. The service can be discovered, composed into larger-scale services, replicated for parallelism and distributed, modified by parametric input. In order for this to be achieved the service needs to be wrapped by rich metadata such that these actions can be automated and not require human intervention. The key question then is what metadata is required for these autonomic actions to be achievable at execute time, and for systems development at system build time.

5.3 Environment

The problem is how to specify, design and construct such systems – or system components – based on SOKUs. Traditional system development technologies are only partly appropriate to this new environment. Instead of the traditional requirements specification, iterative development and delivery of an end-to-end system - which may or may not include preconstructed components – the emphasis is on the development of preconstructed components that can be automatically or semi-automatically composed at both system development time and at execute time. This requires a different kind of analysis of requirements utilising an approach that considers a much wider context than the system being specified in order to optimise the future utilisation of components being developed. In other words the requirement is for the development of generic components that meet the specification of the system currently being developed but also can be re-used in other systems. This approach has been discussed in the past – usually in an object oriented environment with the class concept – but has rarely if ever been achieved. The contention of this paper is that this is because the metadata associated with the components was inadequate for the re-use purpose.

6 METADATA IS THE KEY TECHNOLOGY

6.1 Introduction

Metadata is data about data [7]. An example might be a product tag attached to a product (e.g. a tag attached to a piece of clothing) that is available for sale. The metadata on the product tag tells the end-user (human considering purchasing the article of clothing) data about the article itself – such as the fibres from which it is made, the way it should be cleaned, its size (possibly in different classification schemes such as European, British, American) and maybe style, designer and other useful data. The metadata tag may be attached directly to the garment, or it may appear in a catalogue of clothing articles offered for sale (or, more usually, both). The metadata may be used to make a selection of potentially interesting articles of clothing before the actual articles are inspected, thus improving convenience. Today this concept is widely-used. Much e-commerce is based on B2C (Business to Customer) transactions based on an online catalogue (metadata) of goods offered. One well-known example is www.amazon.com. B2B (Business to Business) is more complex, often involving negotiation to reconcile metadata differences in syntax and semantics including metadata associated with trust and security.

What is metadata to one application may be data to another. For example, an electronic library catalogue card is metadata to a person searching for a book on a particular topic, but data to the catalogue system of the library which will be grouping books in various ways: by author, classification code, shelf position, title – depending on the purpose required.

6.2 Current Availability and Usage

A database schema is a well-known example of metadata. However, it does little more than provide an interface to provide independence between software and data and a naming system for data objects to be used in software. Database schemas do provide information on syntax (structure) at both logical and physical levels but fail to address the semantic (meaning) level.

A URI provides metadata pointing to the actual object of interest. URIs can be rather complex, including not just an internet address but also parameters or even a query.

DC (Dublin Core) is a well-known metadata standard for describing objects, initially designed to describe web pages. It has a set of extensions (qualified DC) which makes interoperation difficult as different developers interpret differently the semantics (and in some cases the syntax) of DC.

There have been various attempts at trust negotiation and rights management/trading using metadata and standards for recording rights have been proposed. The management of rights associated with created works has been discussed extensively [8]. This extends to IPR (intellectual property rights) as understood in this environment. However, the management of rights in a B2B transaction (or a business relationship with multiple transactions e.g. within a supply chain) is more complex. Setting up the relationship involves trust negotiations which in turn require access to conditions of business, background information on the organisation and possibly references concerning the organisation. From this basis an appropriate trust/security policy can be put in place for transactions with the organisation concerned and encoded as metadata which wraps the transactional information in order to ensure the security systems in the organisation operate correctly. A basic trust model is presented in [9] while a more business-oriented approach is that of the TrustCom project [10].

Dictionaries, thesauri, domain ontologies are all required for interpretation of semantics and are used in association with other metadata to support the IT processes, and their users, in understanding and interaction. They are also used for interoperation between heterogeneous systems.

Procedure calls and functional signatures provide some metadata information about the functional properties of a software component. However the information is usually very limited and commonly the information is only encoded as comments within the component. A further problem is that, in general, non-functional properties (such as performance, limitations of use, precision / accuracy etc) are not declared and documented.

O-O classes and KE frames provide metadata again concerning the properties of a software component or software / data component. Herein lies the problem, the confusion between software and data properties and how they should be described or exposed for utilisation by other systems. In general these technologies are being superseded by service-oriented components.

6.3 Kinds of Metadata

It is increasingly accepted that there are several kinds of metadata. The classification proposed (Fig. 3) is gaining wide acceptance and is detailed below.

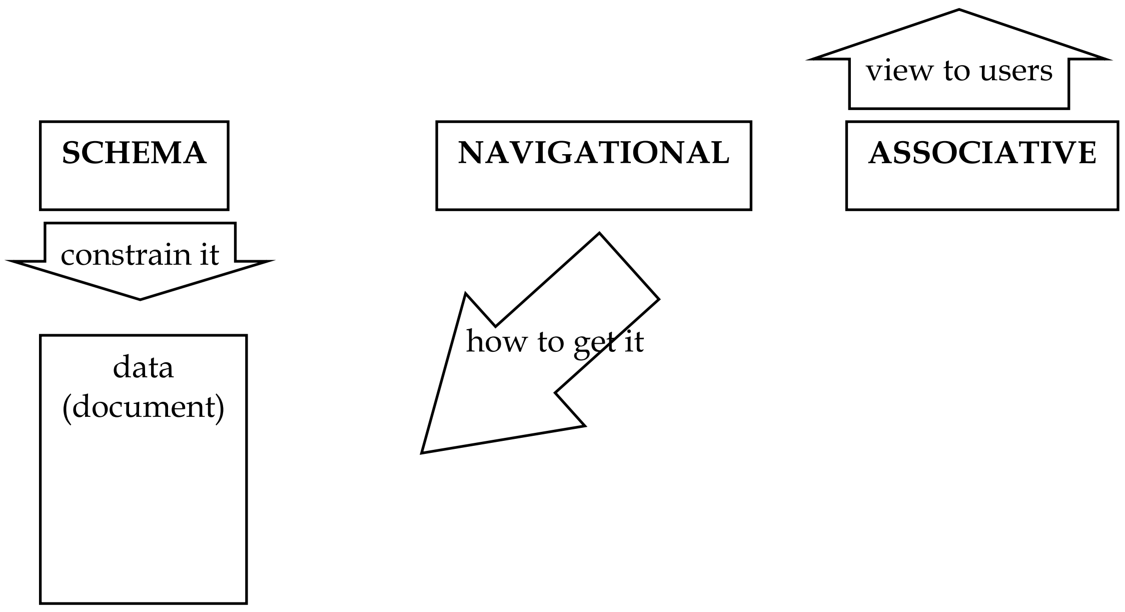

Fig. 3. Metadata Classification

Schema Metadata. Schema metadata constrains the associated data. It defines the intension whereas instances of data are the extension. From the intension a theoretical universal extension can be created, constrained only by the intension. Conversely, any observed instance should be a subset of the theoretical extension and should obey the constraints defined in the intension (schema). One problem with existing schema metadata (e.g. schemas for relational DBMS) is that they lack certain intensional information that is required [11]. Systems for information retrieval based on, e.g. the SGML (Standard Generalised Markup Language) DTD (Document Type Definition) experience similar problems.

It is noticeable that many ad hoc systems for data exchange between systems send with the data instances a schema that is richer than that in conventional DBMS – to assist the software (and people) handling the exchange to utilise the exchanged data to best advantage.

Navigational Metadata. Navigational metadata provides the pathway or routing to the data described by the schema metadata or associative metadata. In the RDF model it is a URL (universal resource locator), or more accurately, a URI (Universal Resource Identifier). With increasing use of databases to store resources, the most common navigational metadata now is a URL with associated query parameters embedded in the string to be used by CGI (Common Gateway Interface) software or proprietary software for a particular DBMS product or DBMS-Webserver software pairing.

The navigational metadata describes only the physical access path. Naturally, associated with a particular URI are other properties such as:
a) security and privacy (e.g. a password required to access the target of the URI);
b) access rights and charges (e.g. does one have to pay to access the resource at the URI target);
c) constraints over traversing the hyperlink mapped by the URI (e.g. the target of the URI is only available if previously a field on a form has been input with a value between 10 and 20). Another example would be the hypermedia equivalent of referential integrity in a relational database;
d) semantics describing the hyperlink such as 'the target resource describes the son of the person described in the origin resource'
However, these properties are best described by associative metadata which then allows more convenient co-processing in context of metadata describing both resources and hyperlinks between them and – if appropriate - events.

Associative Metadata. In the data and information domain associative metadata can describe:
a) a set of data (e.g. a database, a relation (table) or a collection of documents or a retrieved subset). An example would be a description of a dataset collected as part of a scientific mission;
b) an individual instance (record, tuple, document). An example would be a library catalogue record describing a book ;
c) an attribute (column in a table, field in a set of records, named element in a set of documents). An example would be the accuracy / precision of instances of the attribute in a particular scientific experiment ;
d) domain information (e.g. value range) of an attribute. An example would be the range of acceptable values in a numeric field such as the capacity of a car engine or the list of valid values in an enumerated list such as the list of names of car manufacturers;
e) a record / field intersection unique value (i.e. value of one attribute in one instance) This would be used to explain an apparently anomalous value.

In the relationship domain, associative metadata can describe relationships between sets of data e.g. hyperlinks. Associative metadata can – with more flexibility and expressivity than available in e.g. relational database technology or hypermedia document system technology – describe the semantics of a relationship, the constraints, the roles of the entities (objects) involved and additional constraints.

In the process domain, associative metadata can describe (among other things) the functionality of the process, its external interface characteristics, restrictions on utilisation of the process and its performance requirements / characteristics.

In the event domain, associative metadata can describe the event, the temporal constraints associated with it, the other constraints associated with it and actions arising from the event occurring.

Associative metadata can also be personalised: given clear relationships between them that can be resolved automatically and unambiguously, different metadata describing the same base data may be used by different users.

Taking an orthogonal view over these different kinds of information system objects to be described, associative metadata may be classified as follows:
1) descriptive: provides additional information about the object to assist in understanding and using it;
2) restrictive: provides additional information about the object to restrict access to authorised users and is related to security, privacy, access rights, copyright and IPR (Intellectual Property Rights);
3) supportive: a separate and general information resource that can be cross-linked to an individual object to provide additional information e.g. translation to a different language, super- or sub-terms to improve a query – the kind of support provided by a thesaurus or domain ontology;
Most examples of metadata in use today include some components of most of these kinds but neither structured formally nor specified formally so that the metadata tends to be of limited use for automated operations – particularly interoperation – thus requiring additional human interpretation.

6.4 What is Needed Now

The roadmap for moving forward requires several components:

1. a generally agreed understanding of the purposes, uses and needs for metadata in a GRIDs / ambient environment
2. the definition of metadata that is machine understandable as well as machine readable
3. the definition of metadata for description, restriction, correctness, navigational access and system support
4. standardisation of (2) and (3) with widespread deployment and use
5. a process for updating (4)

Clearly (1), (2) and (3) will require more research and development including extensive testing for effectiveness before we can move to (4). I foresee this R&D effort lasting for some years, and keeping active researchers in the area very busy!

7 CONCLUSION

The GRIDs architecture will provide an IT infrastructure to revolutionise and expedite the way in which we do business and achieve leisure. The Ambient Computing environment will revolutionise the way in which the IT infrastructure intersects with our lives, both professional and social. The two architectures in combination will provide the springboard for the greatest advances yet in Information Technology. This can only be achieved by excellent R&D leading to commercial take-up and development of suitable products, to agreed standards, ideally within an environment such as W3C (the World Wide Web Consortium) and/or OGF (Open GRID Forum). The current efforts in GRID computing have moved some way away from metacomputing and towards the architecture described here with the adoption of OGSA (Open Grids Services Architecture). However, there is a general feeling that Next Generation GRID requires an architecture rather like that described here, as reported in the Report of the EC Expert Group on the subject [3]. To develop instances of this architecture will require advanced information systems engineering. The key is advanced, machine-understandable metadata to describe the architectural components.

Acknowledgements
Some of the material presented here has appeared in previous papers by the author. Although the author remains responsible for the content, many of the ideas have come from fruitful discussions not only with the author's own team at CCLRC-RAL but also with many members of the UK science community and the UK Computer Science / Information systems community. The author has also benefited greatly from discussions in the contexts of ERCIM (www.ercim.org), W3C (www.w3.org) and the EC NGG Expert Group.

There is one special acknowledgement. I was working on – or more accurately struggling with – how best to manage the systems development process for advanced information systems applications when I met Arne Sølvberg. We have been colleagues and - I am proud to say - friends for approximately 25 years. We have served together on boards and committees, most recently in the roles of president and vice-president of ERCIM. Arne has been mentor and guide, opponent and partner (in discussions and

debate), supporter and critic. We have worked together on developing a strategy for IT in Europe (and wider) and we have worked together on research projects. Whether in a professional context, or as host (and raconteur) Arne is always excellent company. He has surely been a major influence on my career.

References

[1] Berners-Lee,T; 'Weaving the Web' 256 pp Harper, San Francisco September 1999 ISBN 0062515861
[2] http://purl.oclc.org/2/
[3] www.cordis.lu/ist/grids/index.htm
[4] I Foster and C Kesselman (Eds). The Grid: Blueprint for a New Computing Infrastructure. Morgan-Kauffman 1998
[5] Jeffery, K G: 'An Architecture for Grey Literature in a R&D Context' Proceedings GL'99 (Grey Literature) Conference Washington 2 October 1999 http://www.konbib.nl/greynet/frame4.htm
[6] Original Paper on GRIDs, available at http://www.cclrc.ac.uk/Publications/1433/KnowledgeInformationData20000124.htm
[7] K G Jeffery. 'Metadata': in Brinkkemper,J; Lindencrona,E; Solvberg,A: 'Information Systems Engineering' Springer Verlag, London 2000. ISBN 1-85233-317-0.
[8] http://www.dlib.org/dlib/july98/rust/07rust.html
[9] http://jan.netcomp.monash.edu.au/publications/IC2002_69.pdf[10]
[10] http://www.eu-trustcom.com/
[11] K G Jeffery, E K Hutchinson, J R Kalmus, M D Wilson, W Behrendt, C A Macnee, 'A Model for Heterogeneous Distributed Databases' Proceedings BNCOD12 July 1994; LNCS 826 pp 221-234 Springer-Verlag 1994