Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2006 Proceedings

Americas Conference on Information Systems (AMCIS)

December 2006

Enterprise Software Grid: A Business Grid Service Ontology

David Bell
Brunel University

Sergio de Cesare
Brunel University

Mark Lycett
Brunel University

Enterprise Software Grid: A Business Grid Service Ontology

David Bell
Department of IS, Computing and Mathematics,
Brunel University,
Uxbridge, UK.
david.bell@brunel.ac.uk

Sergio de Cesare
Department of IS, Computing and Mathematics,
Brunel University,
Uxbridge, UK.
Sergio.decesare@brunel.ac.uk

Mark Lycett
Department of IS, Computing and Mathematics,
Brunel University,
Uxbridge, UK.
mark.lycett@brunel.ac.uk

ABSTRACT

Grid services have come to represent the synthesis of web services and grid computing paradigms. Web services provide the means to modularize software is a way that supports their description, discovery and synthesis. Grid computing removes the binding between functional components and specific hardware, enabling software to be deployed dynamically over a network (e.g. intra-, extra- or inter-net). Representation of web service software has tended to be syntactic, lacking fundamental semantic underpinning. Grid services re-enforce the need to address the lack of semantics and enable a commercial semantic grid. The contribution of this paper is the development of a business grid service upper ontology. An upper service ontology that enables business grid services to be described and then related to the grid hosting platform. Tacit knowledge underlying service description and messaging are uncovered. The ontology is derived from and validated using a collection of web services taken from leading investment banks.

Keywords (Required)

Grid Computing, Services, SOA, Ontology, Semantic Grid.

INTRODUCTION

The semantic grid community encompasses two interrelated research areas: grid computing and the semantic web. The community has tended to focus on the knowledge and knowledge tools required to bring semantics to the scientific grid community. The semantic grid vision is summarized in its definition as "an extension of the current Grid in which information and services are given well-defined meaning, better enabling computers and people to work in cooperation" (http://www.semanticgrid.org/vision.html). Commercial organisations, with large software and hardware inventories, would be willing recipients of such a vision in order to benefit from derived flexibility and enterprise software reuse. In order to achieve such benefits, current methods and models require validation and refinement through their application in specialized commercial domains. In this paper the financial services (FS) domain is used.

Literature and practice has provided the grid community with a diverse number of grid constructs, represented in the taxonomies that exist (Krauter, Buyya et al. 2002, Yeo, Buyya , Yu, Buyya 2005), and often include only homogenized shared resources such as CPUs. For example, the classic grid typically contains resources such as databases, cluster nodes and devices. Transferring classic grid computing to a business domain has tended to involve transferring existing constructs. This had led to constrained application of the grid in a business setting with only computation applications benefiting. Another symptom of constraint is a singular focus on grid standards. This research aims to support enterprise software grids through the identification of appropriate conceptual models. It would be a mistake to accept current resource models when the entities being re-used are not of existing grid types. Two complementary research strategies are executed in order to produce a balanced, FS grounded grid service semantics. Firstly, a top down approach is used to classify a commercial software inventory of software components using an existing grid computing taxonomy. Second and counteracting any bias embedded in the existing taxonomy, bottom-up developmental analysis is carried out. Three popular grid development

approaches – command line, portal and web service – are used to replicate a typical risk analysis use case from the FS domain. The contribution of this paper is the development of a business grid service upper ontology.

The paper is structured as follows. Initially, a summary of existing types of grid computing is presented. This is followed by the primary research, with sections covering the evolution of a grid taxonomy and prototype service development. Finally, the paper concludes with an evaluation of the research artifacts, aggregated to form a business grid service ontology, and a summary.

BUSINESS GRID SERVICES

Grid computing, as documented by the literature, can take many forms. Some common examples include:

- Load balancing – Condor, Platform LSF, IBM LoadLeveler
- Cycle stealing – Seti@Home, BOINC
- Service Oriented Middleware – Globus, Sun Grid, Web Services WS
- On-demand Computing – IBM WebSphere
- Pervasive Computing – JXTA P2P
- Utility Computing – IBM, HP, SUN

Variation, identified by Foster (Foster 2002), demonstrates a lack of coherence in current constructs and motivates work to identify grid constructs relevant to business service grids – uncovering business grid service semantics. Foster is correct to reclassify some of the above systems as grid resources not grid systems as much of the existing load balancing or hosting platforms are required components of a grid, but not a grid in their own right. However, the aim of this research is enterprise system reuse and should focus on the resources and platforms. This is achieved by extending a widely cited grid system taxonomy (Krauter, Buyya et al. 2002) - representing the platform - and linking it to resource centric development approaches. Clear and explicit constructs relating to business grid computing will enable the research to achieve better designs as they "provide the language in which problems and solutions are defined and communicated" (Hevner, March 2003 p. 111). The refinement of constructs is a natural part of the design research approach, and result in valid research contribution if a young domain is given greater clarity. Business application software exploration is likely to involve service oriented approaches where services may communicate with or be dependant on each other. In order to support this, a technocratic approach to knowledge management is required – focusing on systems and engineering (Earl 2001)– exploring the use of knowledge bases and flows to better identify software components. In order to develop this line of research, constructs are required to describe the problem and solutions.

Two research strategies are undertaken in order to develop the service semantics for business grid services: (1) A mapping of software from three FS Investment Banks to an existing grid taxonomy and (2) development of prototype business grid services using current approaches. The aim being that on linking the top-down taxonomy extension to the bottom-up development strategies, a practical semantic model can be achieved. It should also be noted that the starting point is the commercial software inventory and not existing semantic models. Purely applying OWL-S (Martin, Paolucci et al. 2005) or WSMO (Lara, Roman et al. 2004) would not allow the research to uncover some of the subtle transformations required and then reflected in the ontology.

EXTENDED GRID TYPE TAXONOMY

Categorisation of grid resource types with grid architecture may yield some new constructs. Reverting to a top down approach, classifying enterprise software as grid resources requires suitable enterprise software for analysis. Investment banking IB is the chosen domain because of the global reach and continually changing strategy and structure (Huang, Newell et al. 2003, Jones 2005, Peinl 2000, Schamp, Rentmeister et al. 2004). A taxonomy of grid systems, extending on from work by Krauter et al. (Krauter, Buyya et al. 2002), provides the design exercise with a grid taxonomy from which to develop grid resources association – relating enterprise software (represented by WSDL definitions) to grid types. Definition of scope and underlying concept and constructs (design research process) rely on a clear understanding of what grid computing is and in particular what are the resources under management. Grid types are presented in Figure **1**1.

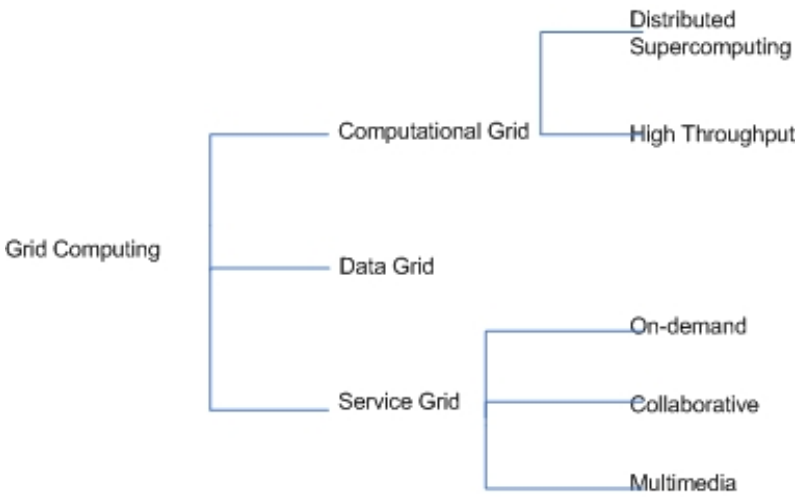

Figure 1: Grid Taxonomy (Krauter, Buyya et al. 2002)

The taxonomy provides a hierarchy of types for grid architecture. Each type represents a particular grid topology at a macro level. The discrete components that make up such a grid, its resources, can be inferred but are not explicit. The types may be described as follows:

§ Computational grids: Provide higher aggregated computational capacity (Krauter, Buyya et al. 2002) – examples include *distributed supercomputing* where reduced job completion time is achieved through parallelism and *high throughput* where multiple, dependent jobs run faster through distributions across multiple hosts.

§ Data grid: This is an infrastructure for organizing, cataloguing, managing and accessing distributed data sets such as libraries or data warehouses.

§ Service grid*:* enables the flexible provisioning of services over a grid network, as opposed to single machines. Further categorization of service grid covers sporadic access to services "on-demand", support for collaborative workgroups of humans and systems, and distributed multimedia application. It is on-demand service grids that best describes the area under investigation. Existing enterprise systems are recast in a grid environment in order to better access and use their valuable software inventory.

On reflection, such grid types seem appropriate. However, when considering that a particular grid type is selected for the hosting of enterprise components; such a type system is too general. Consequently, it is the aim of the first part of this research to validate and refine the taxonomy using a current enterprise software inventory.

BUSSINESS SOFTWARE MAPPING

The grid taxonomy provides a basis for categorizing real-world software components. Three organisations are used to source such software artifacts – described using WSDL. The artifacts, more specifically the WSDL operations, are used as representative components of a distributed enterprise environment. A framework, presented in Table 1, is used to categorise each software artefact using the grid taxonomy. The process is covered later in this section.

The organizations under analysis are multi-national investment banks – both European and Global top-tier. The organizations are summarized below in order to give a context to this and later analysis.

FS ORGANISATION SOFTWARE INVENTORIES

Each of the banking inventories being analyzed use web services as a development approach. They each have a different sub-domain – treasury TREAS, derivatives LOBANK and fixed income HIBANK – as well as technical approach. The treasury organization uses simple types, whereas the derivatives organisation uses complex schema when describing their

Activities	Description	Input Artifacts	Output Artifacts
Service Identification	A service description is broken down into its fundamental parts (e.g., name, input and output parameters). Each part is interpreted in order to identify relations to grid types	Web service descriptions (e.g., WSDL code)	Grid Association Models
Process Analysis	The association model, exposing implied process, are mapped to Grid resource types that best represent their operation	Grid Association Models Grid Taxonomy	Grid Mapping
Service Categorisation	Grid Mappings are organized into a taxonomy that encompasses the interpreted software artifacts	Grid Mapping	Business Grid Resource Constructs

Table 1: Mapping business artifact to Grid concepts ("association framework")

software. The type of functionality found in the banking organisations is as follows:

- § Extracting vectors of cash flows for a particular trade or book of trades
- § Executing trades of differing type
- § Extracting trades that have been executed between a specified data range
- § Calculating the profit and loss of particular trades
- § Valuing trades with specific market data scenarios
- § Undertaking various business processes, such as novation, cancellation, rate fixing
- § Aggregating trading totals into groupings based on the underlying trading asset (termed the position)

In order to fully explore the issues associated with the integration of enterprise systems, working examples of such systems, taken from the above organisations, are analysed. The association framework is used interpret explicit and implicit concepts – derived from WSDL Operations.

Table **2** summarizes the number of web services used by each of the organisations specific projects. The number of web services reflects the number of operations defined within the WSDL description. The number of elements (w*eb service elements*) identified at this initial stage are the service name, the parameter names and the output name (again all resident in the WSDL document). HIBANK elements also included a number of external schemas utilised within the WSDL Operations.

Organisation	#Web Services
TREAS	2
HIBANK	39
LOBANK	18

Table 2: Research Input Artifacts (Web Service Operations)

Business Software Mapping Approach

The aim of mapping current software components to existing grid constructs is to develop a conceptual understanding of business grid services. In so doing, current constructs are validated within a business software context. Additionally, such a mapping exercise aims to identify new constructs that are required to describe and discover components when conceptualized as grid resources. The mapping exercise follows the association framework to systematically analyse software components: interpreting, analysing and associating constructs from described services to grid constructs. The components under analysis

are web services, or more specifically WSDL operations (see Figure 2). The operation provides a starting point for semantic analysis. The web service elements used as an input to the research are: (1) The operation name <wsdl:operation name> (2) The input parameters <wsdl:input message> and (3) the result <wsdl:output message>. It is the operation that is analyzed within this paper.

```
<wsdl:operation name="priceTrade">
  <wsdl:input message="Trade"  name="Trade"/>
  <wsdl:output message="NPV"   name="NPV"/>
</wsdl:operation>
```

Figure 2: WSDL Operation example

Interpretation of the enterprise web documents (WSDL and XML Schema) yields many new concepts, exposing previously implied knowledge as new explicit concepts. In order to undertake this task, domain knowledge is required in order to model service concepts with respect to the domain (see Figure 3). The association process has two steps: (1) additional domain concepts are identified using *refers-to* relations and (2) identified concepts are then categorized as particular grid resource types.

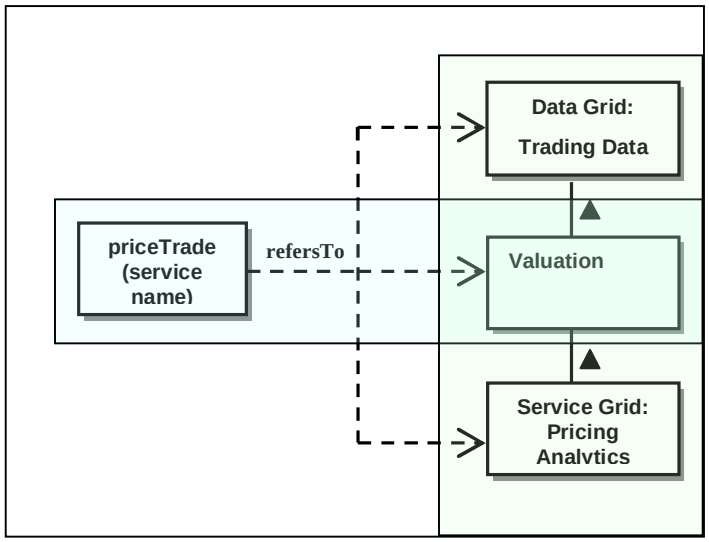

Figure 3: Example Grid Association Model

Fifty eight services across the three organisations are analyzed. The production of grid association models are simplified through the use of *triples* to describe commitment; with the usual *subject, property (verb), and object* format. The use of triples evolved through simplification of modeling process. Essentially, the modeling is uncovering triples graphically and provides a natural way of identifying how objects are grounded within a domain. Affecting such a choice is the current practice of using RDF – a language which encodes triples (Berners-Lee, Hendler et al. 2001). Following the framework involved sequential interpretation of each WSDL operation. The benefit of such an approach is that common properties and objects are identified in line with preceding analysis. In contrast, identifying objects first would not benefit from the hints embedded in the operational declaration (parameters, comments, linked words). Subjects are concepts interpreted from WSDL – the web service elements. Objects are the new concepts exposed while fully describing each subject. Properties are the binary relations between subject and property. It became apparent as the interpretation unfolded, and within the example above, that two triple types result. The *Resource triples (horizontally shaded)* results from the first relation to the domain. *Resource Dependency Interpretation triples (vertically shaded)* result from the analysis of the domain object in relation to the wider grid domain. The use of triples enabled ontology web documents to be formed and visualized using our own simple transformations tools. A summary of the types of grid resources identified is provided in Table 3. Traditional (or Classic) grid resources are represented by Computational (0%) and Data grid resource (37%).

Group	Number (%)
WS Operations	59 (100%)
Data Grid	22 (37%)
Service Grid	37 (63%)
Computational Grid	0 (0%)
Service Grid with data dependency	31 (53%)
Service Grid with Computational dependency	3 (5%)

Table 3: Categorization Summary

Not surprisingly, the majority of the software maps onto service grid as WS components being analyzed. The study does however highlight the dependency that many of the services have on related data. New domain concepts are uncovered during the analysis and require explicit definition if they are to be used to discover and describe services using such concepts. Analysis of commercial software inventories provides a basis for extending an existing taxonomy. Basing such extension on IB software inventories has resulted in an IB specific taxonomy. The development of such taxonomy is done in order to: (a) ground the taxonomy in service and domain objects and (b) provide semantics relating software component to their hosting environment. Consequently, the taxonomy provides a link between two resource groups – the enterprise software and the hosting service. Practically, such a association is directed by the current hosting service, the code and the WSDL description. Development of the taxonomy is carried out by creating subclasses for the identified resource types. The extended taxonomy is presented in Figure 4.

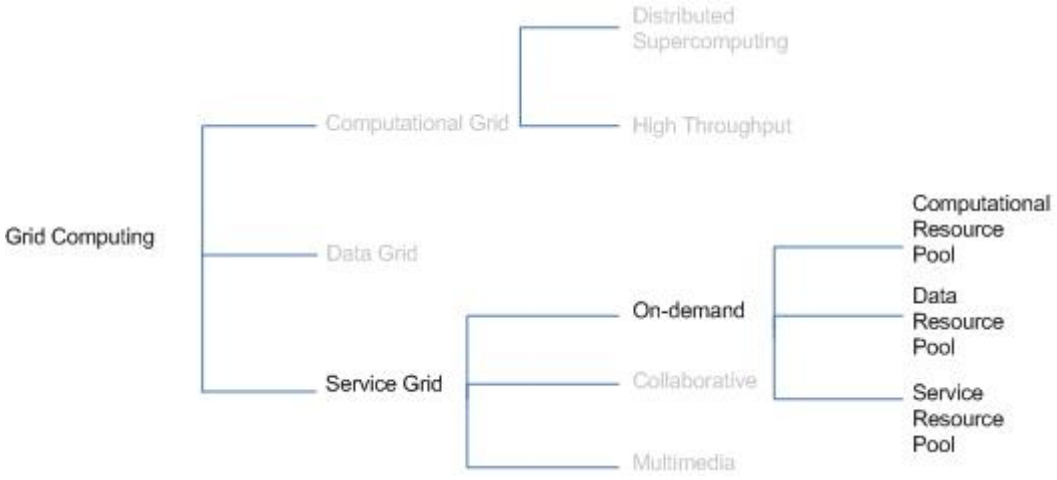

Figure 4: Extended Service Grid Taxonomy (based on Krauter, Buyya et al. 2002)

Extension of a recognised Grid taxonomy with additional constructs forms a model of sorts, with methods being the process undertaken to extend the taxonomy. The justification for such extension is grounded in and a function of the commercial software inventory. Continuing with the example, the taxonomy is able to describe a "pricing analytics service resource pool". Consequently, a granular business grid service hosting environment is achievable. The ability to better describe the grid system does not however address the more practical issues around the actual resources being deployed on such grids – specifically what are grid resources in a business system reuse context. Consequently, a second, bottom up research path is undertaken - developing practical IB business grid service instantiations. Instantiations realizing the same set of resource types, using differing patterns of development, enable the research to uncover some development related issues.

BUSINESS GRID SERVICE DESIGN ALTERNATIVES

Both the development and use of particular grid resources is undertaken for specific resource types and development approaches. Three typical approaches are chosen to develop a group of different resource types. The rationale for choosing the three is that they each represent an existing community and complexity level. Before describing the scenario, the aforementioned popular approaches are summarised:

- § Executable Based Grid Resources (EXE)**:** Software components are developed as discrete executables that can be deployed and executed on the grid. The interaction is typically carried out with a command line interface (e.g. Globus GT2-4).
- § Portal based Grid Resources (PORT): Software, in executable form, is deployed and executed using a web based portal environment. For this research, the P-Grade/NGS portal is chosen (itself based on GridSphere (GridSphere 2005)) because it includes a directed acyclic graph (DAG) workflow component that provides automatic choreography of input and output files between executables.
- § Web Service Based Grid Resources (WSG): Software components are developed as stateful web services using the WSRF standard. The services are then deployed and executed within an application server environment (e.g. Globus GT 4 Core Java APIs).

Continuing with an FS focus a suitable scenario is chosen that composes a set of distinct resource types. The scenario chosen is the risk management use case (typical of many banking institutions)(CDDLM 2003). Portfolios of trades are subjected to differing market conditions to determine the change in value. These market changes are typical of a parameter study where a range of shifts to existing market measures are passed to the valuation service. The execution of such a process involves three stages: (1) Trades and market data are gathered (the inputs to the pricing process), (2) The trades and a shifted list of market data (often in yield curve form) are passed to valuation service for pricing and (3) the results are aggregated to form both summary and detailed reporting. The use of this particular scenario is natural as it is already commonly carried out using cluster computing solutions in the commercial environment. It should be noted that the type of trades under analysis determine which market data and valuation service are required.

The fist step is the development of prototype modules. In this experiment, market data, trade data, risk analysis and reporting modules were developed in C. Key characteristics of each module are that: they accept and generate XML data files, the state of the process is embedded within the module and each module has no dependency on library code. This classic grid model relies on discrete executable software components, and input and output data files, being deployed and executed. Two critical issues arise when adopting this approach – namely binary execution compatibility and dependency. Executables must be appropriate for the environment to which they are to be deployed. For example, a C program compiled for one chipset cannot be executed on another. A true virtual environment, with brokers supporting these constraints, will hide such issues. However, the Grid of early 2006 often requires the user to specify a particular host and therefore should be an included instantiation. Dependency is also critical. The executable may require libraries or the like to support its execution and these may not be resident on the host. This again can be added to the broker constraint list or included in the deployment process. The broker could choose only those hosts that support MATLAB or a particular JVM version. Alternatively, if a particular library file is required, it can be deployed prior to execution. Thus, even during the design of a classic grid, *executabletype* and *dependencies* are added to the resource characteristics. The Globus middleware is used to execute the modules and transfer the input and output files.

A second, user centric, approach to utilizing grid computing is that of the grid portal. Several portals have been produced within an academic setting. Of these, National Grid Service (NGS) Portal, Triana and P-Grade were considered as candidates. P-Grade was chosen because it included a workflow component (DAGMan). DAGman, from the Condor Project, uses directed acyclic graphs to describe the flow between nodes on the grid. NGS Portal was deemed too simplistic and Triana did not utilize the NGS grid. Utilizing a portal environment enables investigation into characteristics that were deemed missing from basic command line grid interfaces. Two areas of portal use that are considered, as part of this research phase, are the selection of grid resources and execution of a predefined workflow. This is where resource description and selection is carried out. The utilization of a portal relies on the combination of software (again in executable from), data and hardware hosts. The portal itself has portlets that provide an interface to information services such a MDS and certificates downloaded from the MyProxy server. Although useful, the portlets of this type only make the scenario setup less troublesome. The existing executables from the EXE development are chosen to be executed on an available node of the grid. Symptomatic with the current grid, this choice is fixed. The user is given a large number of nodes to choose from, and it is the user that makes the binding between executable, data and node. This can be simplified by the use of broker (such as

LGS), but still requires an abstract description of the hosting environment required. The workflow is then gradually built by adding appropriate jobs. The jobs are linked together using ports. These ports depict gridFTP file transfers between nodes.

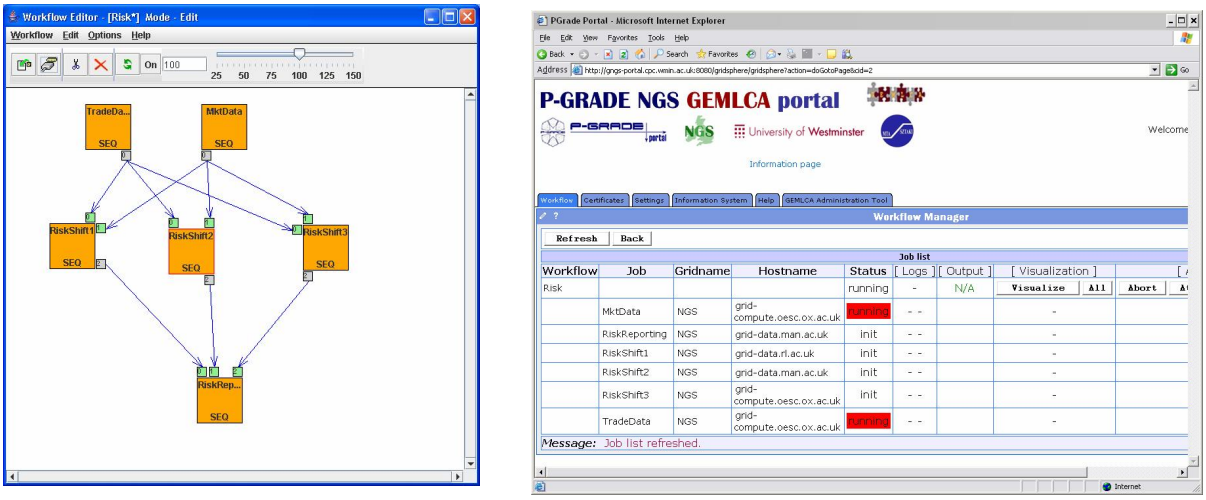

Figure 5: Portal Development

With a workflow defined and saved, it can then be executed. It common with most portal environment, the execution can be monitored and visualized (as seem in Figure 5).

In common with the EXE development, the focus of the portal is on the three key constructs – hosts, executables and data (HED) – and their interdependency. Forcing users to understand the relationships between each of these constructs may result in the grid system being used by grid developers – and not the scientists or business user to which it was aimed. It may be inferred from this that the scientist cannot conceptualize problems using the constructs being provided. This is a critical issue because the business grid contains service based resources that are more heterogeneous than the science grids – due in part to both a varied user community and a varied software inventory. The design must address this issue – allowing the user to select business grid resources using domain language and without the need to understand the underlying technical bindings.

In contrast to the EXE development, PORT development, through the DAGMan workflow, highlight the specific dependency between grid resources within the workflow. This is traditionally an output file generated by one host being required by an executable on another host. This dependency is a particular state that another resource finds itself in. Rather than directly transferring this approach to the business grid, a fuller understanding of this state should be investigated. The reason for this is that for service based resources, state moves beyond being a particular output file. Intermediate or final states from services may be interchangeable in a more synergistic manner. Differing services may calibrate input data in the same way and this intermediate state may provide a more effective resource. Finally, a third development alternative is investigated – web service grid development. Developing the scenario using WSRF takes a more resource, as opposed to host, centric approach. It is apparent that the grid computing development is in a state of transition - moving from a host-executable-data HED to service views of grid resources. This is apparent when contrasting current large production grids to the grid standards work. Many production grids are still using the Globus GT2 middleware or Condor. On the other hand, the grid standards are being integrated with web services to form WSRF, a resource framework based on services. Attempting a development based on WSRF forces the risk management resources to be re-factored with a service interface.

We now assume a business grid resource in WSRF terms (as opposed to a HED model). The reasoning being that WSRF provides: (1) a factory pattern for software use and (2) introspective abilities that allow the internal state of the resource to be queried by the user. Heterogeneous software services, with heterogeneous internal properties, do not lend themselves to direct transformation to HED. The development of the risk management scenario using WSRF opens two alternative transformation approaches. Put simply, either the host or executable inventory can be made available using WSRF. The association modeling directs this choice toward software as service associations were in majority. The software conceptualization moves from a static focus on functional signature to one of dual utility – combining the benefits of mobility and the value of particular service states having value. Business grid services become factories of operational software with varied state. Figure 6 highlights this in a high level view of stateful software services deployed and executing on hosts. The re-engineering of software developed and used for EXE and PORT development involved the placement of code behind a web service interface (see Figure 6):

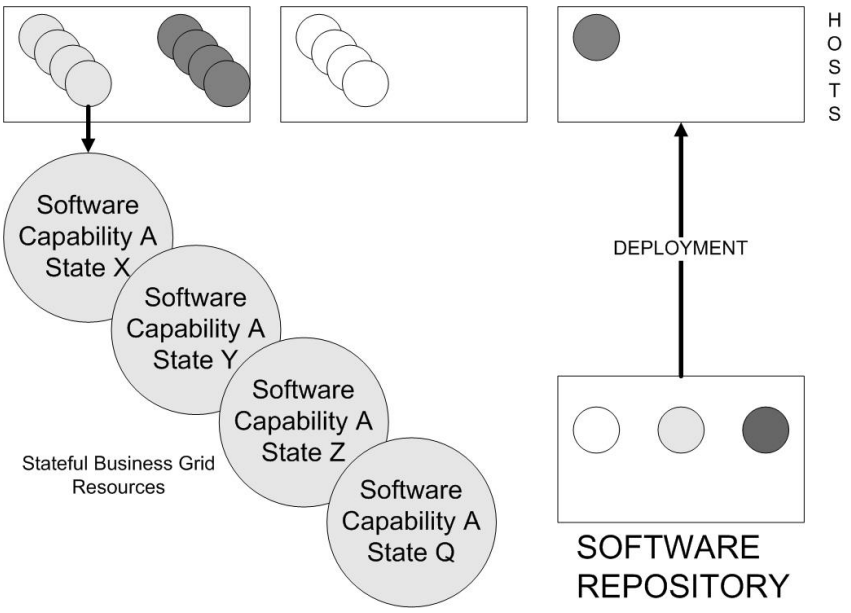

Figure 6: Service Grid – A factory of stateful resources

Software provides the resources for a workable business grid where such a service can be executed, becoming a resource with a particular execution state. The two approaches to a WSRF grid highlight a need for a two-phase discovery process: (1) discovery of the services required and (2) discovery of the hosts where such services can be deployed if not already. Following the risk management scenario the *MarketData* service would load market indicators (representing a particular market M) and then build a yield curve of discount factors. If another user requires this service, they have the option of calling the stateful resource and use its calibrated results or call the service that will then create another resource. Obviously, the existing resource will only be reused if the market M is required by the second user. The consequence of this business grid approach to the software inventory is that over time a grid of available software components will be available for re-use.

EVALUATION

Three key finding can be taken from the research: (1) Software grids have two layers of resources – the software and the host environment in which they are deployed, (2) domain ontologies are critical if the software grid is to be able to expose, aggregate and select service state in the same way that CPU usage is used in the classic grid model and (3) the newer web service grid best support the reuse of enterprise software. Translating these finding into the required semantic for business grid services moves service ontologies away from the service centric approaches of OWL-S and WSMO to a more holistic upper ontology. Relationships that bridge resource layers need to be made explicit, for each software resource and not just the application as a whole (as seen in (Zhang, Zhou et al. 2002)). High level grid service ontology that reflects these finding is presented in Figure 7. Its formation is the result of an effective two strategy derivation process – one that provides both ontological artifacts and their inter-relationships. Identification of two dependent layers of grid resources – the business grid services and the software hosting services – provides a novel approach to grid architecture. Extending the taxonomy provides a bridge between the two layers. The ontology provides a basis for additional research into methods for the population and linkage between underlying ontological artifacts.

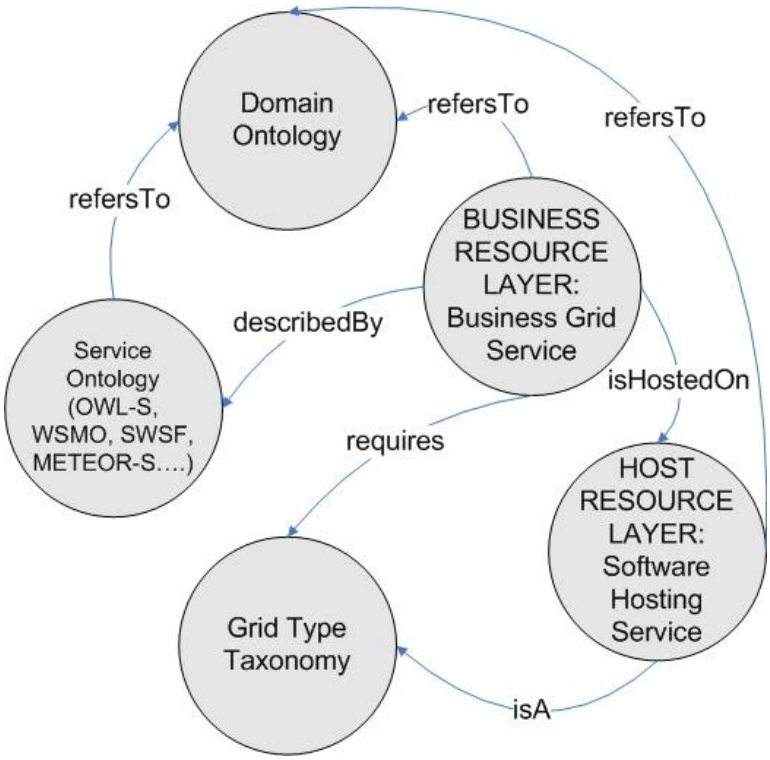

Figure 7: **Business Grid Service Ontology**

The outcome of WSG development work demonstrates how business software capabilities are easily exposed as grid services – requiring only additional state management identification and coding. The assumption can now be made that a corporate inventory can be made available using grid infrastructure and result in a topology of resource factories managing software capabilities – business grid services. Within WSG we have seen a very different grid to that of the HED based EXE and PORT grids. The focus has changed from how we relate objects to the environment to understating internal characteristics of the objects themselves**.** This fact is important as it recognizes the need to accurately reference the domain ontology. Although, considering the software inventory being deployed as a business grid will likely affect any decision. An inventory of executables will advance the EXE and PORT approach. An inventory of software that, once executing, becomes more valuable when in particular state, advances the case for re-factoring with WSRF.

CONCLUSION

The paper presents research carried out to derive a holistic business grid service upper ontology. Two research approaches are undertaken to uncover the semantics of enterprise software components recast as grid resources: (1) A top-down extension of an existing grid taxonomy and (2) practical development of prototypes that test the extended taxonomy and relate it to business grid resources. Resulting from the research are high level business grid service semantics. Applying such a model to an enterprise software inventory provides the means to better utilize grid computing in a commercial setting. Proposing a semantic model of this type, and grounding it in a financial services software inventory, enables the current and future semantic grid tools and techniques to be applied in a business context. It is also clear that a more holistic approach to ontology engineering is required to support business software grids – with explicit engineering and linking to business domain and hosting ontologies. A research contribution is made by (1) identifying the semantic differences between scientific (typically computational) and business software grids and (2) producing and ontology that encapsulate service, hosting and domain knowledge (extending current WS ontology).

REFERENCES

1. Berners-Lee, T., Hendler, J. and Lassila, O., 2001. The Semantic Web. *Scientific American*, **284**(5), pp. 34-43.
2. CDDLM, 2003. Configuration description, deployment, and lifecycle management - foundation document. Available http://www.ggf.org
3. Earl, M., 2001. Knowledge management strategies: Toward a taxonomy. *Journal of Management Information Systems*, **18**(1), pp. 215.
4. Foster, I., 2002. What is the Grid? A three point checklist. *Grid Today*, **1**(6).
5. GridSphere, 2005. Available: http://www.gridsphere.org
6. Hevner, A.R. and March, S.T., 2003. The Information Systems Research Cycle. *Computer*, **36**(11), pp. 111-113.
7. Huang, J.C., Newell, S., Galliers, R.D. and Pan, S.L., 2003. Dangerous liaisons? Component-based development and organizational subcultures. *IEEE Transactions on Engineering Management*, **50**(1), pp. 89-99.
8. Jones, A., 2005. Truly global corporations? Theorizing 'organizational globalization' in advanced business-services. *Journal of Economic Geography*, **5**(2), pp. 177-200.
9. Krauter, K., Buyya, R. and Maheswaran, M., 2002. A taxonomy and survey of grid resource management systems for distributed computing. *Software-Practice & Experience*, **32**(2), pp. 135-164.
10. Lara, R., Roman, D., Polleres, A. and Fensel, D., 2004. A Conceptual Comparison of WSMO and OWL-S, *Web Services: European Conference, ECOWS 2004*, September 2004, pp254-269.
11. Martin, D., PaoluccI, M., Mcilraith, S., Burstein, M., Mcdermott, D., Mcguinness, D., Parsia, B., Payne, T., Sabou, M., SolankI, M., Srinivasan, N. and Sycara, K., 2005. Bringing semantics to web services: The OWL-S approach. *LECTURE NOTES IN COMPUTER SCIENCE*. BERLIN: SPRINGER-VERLAG BERLIN, pp. 26-42.
12. Peinl, P., 2000. Distribution, replication, parallelism, and efficiency issues in a large-scale online/real-time information system for foreign exchange trading. *Euro-Par 2000 Parallel Processing, Proceedings*, **1900**, pp. 451-454.
13. Schamp, E.W., Rentmeister, B. and Lo, V., 2004. Dimensions of proximity in knowledge-based networks: The cases of investment banking and automobile design. *European Planning Studies*, **12**(5), pp. 607-624.
14. Zhang, L., Zhou, Q. and Chung, J., 2002-last update, developing grid computing applications, part 2 [Homepage of IBM Develper Works], [Online]. Available: http://www-128.ibm.com/developerworks/grid/library/gr-grid2/.
15. Yeo, C.S. and Buyya, R., A taxonomy of market-based resource management systems for utility-driven cluster computing. *Software: Practice and Experience*.
16. Yu, J.D. and Buyya, R.D., 2005. A Taxonomy of Workflow Management Systems for Grid Computing. *Journal of Grid Computing*, **3**(3), pp. 171-200.