# Enterprise Application Reuse: Semantic Discovery of Business Grid Services

David Bell[(1)], Simone A. Ludwig[(2)], Mark Lycett[(1)]

Corresponding author: David Bell (David.Bell@brunel.ac.uk)

[(1)] Dept. of Information Systems and Computing
Brunel University
Uxbridge, Middlesex
UB8 3PH, United Kingdom

e-mail: firstname.lastname@brunel.ac.uk

[(2)] School of Computer Science
Cardiff University
Cardiff,
CF10 3XQ, Wales,
United Kingdom.

e-mail: Simone.Ludwig@cs.cardiff.ac.uk

# Enterprise Application Reuse: Semantic Discovery of Business Grid Services

David Bell[1], Simone A. Ludwig[2] and Mark Lycett[1]
[1]*Department of Information Systems and Computing, Brunel University, UK*
[2]*School of Computer Science, Cardiff University, UK*
*David.Bell@brunel.ac.uk, Simone.Ludwig@cs.cardiff.ac.uk and Mark.Lycett@brunel.ac.uk*

## Abstract

*Web services have emerged as a prominent paradigm for the development of distributed software systems as they provide the potential for software to be modularized in a way that functionality can be described, discovered and deployed in a platform independent manner over a network (e.g., intranets, extranets and the Internet). This paper examines an extension of this paradigm to encompass 'Grid Services', which enables software capabilities to be recast with an operational focus and support a heterogeneous mix of business software and data, termed a Business Grid - "the grid of semantic services". The current industrial representation of services is predominantly syntactic however, lacking the fundamental semantic underpinnings required to fulfill the goals of any semantically-oriented Grid. Consequently, the use of semantic technology in support of business software heterogeneity is investigated as a likely tool to support a diverse and distributed software inventory and user. Service discovery architecture is therefore developed that is (a) distributed in form, (2) supports distributed service knowledge and (3) automatically extends service knowledge (as greater descriptive precision is inferred from the operating application system). This discovery engine is used to execute several real-word scenarios in order to develop and test a framework for engineering such grid service knowledge. The examples presented comprise software components taken from a group of Investment Banking systems. Resulting from the research is a framework for engineering service*

*knowledge from operational enterprise systems for the purposes of service selection and subsequent reuse.*

**Keywords**: Semantic Grid, ontological modeling, service discovery, semantic search.

## 1. Introduction

The Semantic Grid is an extension of the current Grid where information is given well-defined meaning, better enabling computers and people to work in cooperation. The aim of the Semantic Grid is to have data and services on the Grid defined and linked in a way that it can be used for more effective discovery, automation, integration, and reuse across various applications. Work proceeds on the basis that the Grid can reach its full potential if it becomes a place where data and services can be shared and processed by automated tools as well as by people (adapted from the Semantic Web Vision [1]). This vision has been applied to many existing e-Science applications that utilize workflow and discovery techniques to identify and synthesize the required resources. Unsurprisingly, business organizations housing a large inventory of data and services are willing beneficiaries of such a vision.

The crux of achieving this vision lies in successfully converging the web service and grid 'paradigms'. Web services offer software component reuse, but are unable to offer the potential benefits achieved from reusing part completed processes – essentially because they are stateless in nature. Conversely, current grid approaches are stateful: - they conceptualize software as a dynamic, deployable component without recognizing the value of any embedded business state (a software component is simply a resource which is tied to a particular machine). The combination of web service and grid paradigms is a potentially fruitful one however, and is currently being touted as the Semantic Grid. The paper presents a service knowledge engineering framework - an application of the Semantic Grid - as an infrastructure for reusing existing enterprise software components. Achieving this reuse of business software is dependant on common meaning being attributed to these software components and their characteristics – service semantics. In addition, the process of reuse requires activities related to *finding* the artifacts to reuse and *synthesizing* the set of artifacts as a coherent whole. This paper investigates the former of these needs through the development of an application prototype grounded in three Investment Banking systems. The area of particular interest is the application of semantic search to grid enabled enterprise software components – evolving a theoretical framework to direct how this activity is architected. Enterprise applications deployed through the global

enterprise often have an implied state that if available for reuse become important. Taking a trade valuation process, for example, significant calibration activity is required in order to determine market value. This calibration involves the loading of market data and building memory-based structures that support generic trade valuation algorithms. Recognizing, describing, discovering and utilizing this calibrated stateful business component provide a novel use of the Semantic Grid. For the remainder of this paper, components such as methods, procedures or scripts providing the public interface that users access are termed *capabilities*. The grid enabled capabilities residing in enterprise applications are referred to as *Business Grid Services.* Exposing business grid services provides the enterprise with a dynamic inventory that can be discovered and re-used by requesting applications – and in particular, those of an "on-demand" nature.

The structure of the paper is as follows. Section 2 provides an example context for the use of business grid services from a software and architecture perspective. Section 3 discusses related literature: While the literature specific to business grid computing is limited, this is addressed by covering the interface between semantic search in grid computing and the more mainstream Web services ontology research. Section 4 then covers the research design and includes a design research framework. Section 5 covers the discovery architecture, implementation and quantitative evaluation of the ontological artifacts using real-world scenarios. Section 6 details the architectural framework, a result of the literature, design, development and evaluation. The paper concludes with an evaluation of design artifacts and summary of the framework and its implications in Section 7 and 8.

## 2. Business Grid Service Context

For reference, the meaning of the core terminology used within this paper is as follows:

- Grid Services are extended web services that provide a programming environment for stateful services with asynchronous messaging feedback. Implied with the term is a notion of flexible deployment and use of components within a system context. The conceptualization of software capabilities as grid services is detailed later in this section and the feedback messaging of resource properties is used as a basis for the ontology extension that is covered in 5.1.2.

- Semantic Search involves the use of subsumption reasoning to replace classes with parent or child classes on the same branch. The semantic structure is described within a domain or service ontology in order to explicitly describe objects that exist within the business domain of study.
- Concepts are business terms that is used to describe (in precise or abstract form) a "thing" that exists within the domain. Concepts used in independent locations, domain specialisms or contexts are often different.

Combining web service and grid paradigms provides the infrastructure for exposing business capabilities for re-use and novel re-configuration. Capabilities within the enterprise application are re-conceptualized as a grid of operational software resources. These capabilities are then available for re-use as a service in either vanilla or specialized state (for example, a pricing capability that is calibrated during previous execution, by another user with particular market data such as Dollar Interest Rates, can be used by the new user request). The logical next step, with a service oriented inventory, is to identify practical methods for describing and discovering these capabilities and support their reuse. Figure 1 provides a simplified picture of this recasting of the enterprise inventory. Capabilities are deployed over the network as requests for use are made. Once deployed, the capability is used in the normal manner. As more users exploit each capability, a deployed group of operational services are available for re-use. Within figure 1, four capabilities titled C have been deployed and used. A future user can now request Capability C in State Z if this has a particular business value over deploying a new capability C. Put another way, Grid services result in a number of instantiations of the same service with independent state. This business state is proposed as an important currency for reuse in this context. Subsequently, client applications can then bind to a new or existing instantiation.

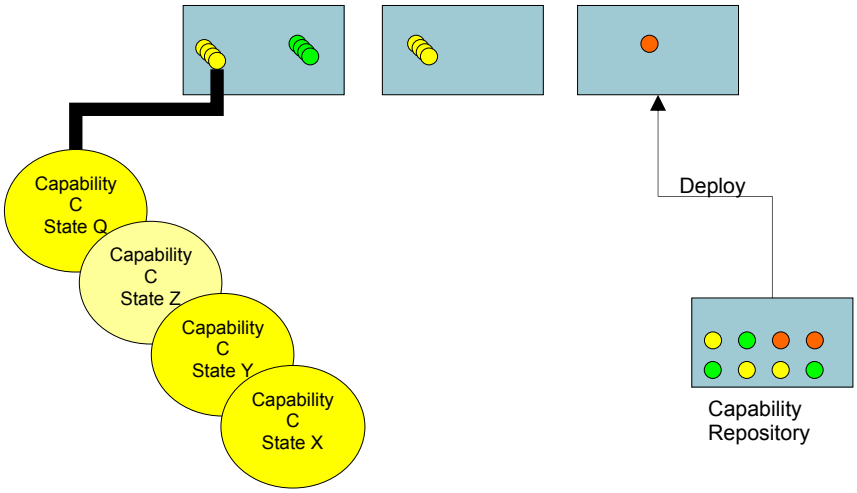


**Figure 1. Business Grid - Stateful Business Capabilities**

In a typical enterprise setting, capabilities are distributed around the network, providing grid service access to existing enterprise capabilities. In an extreme case, a particular user may combine capabilities from applications residing in the organizational centre, a regional hub and/or a specific country without knowing the capabilities exist a priori (highlighted in Figure 2).

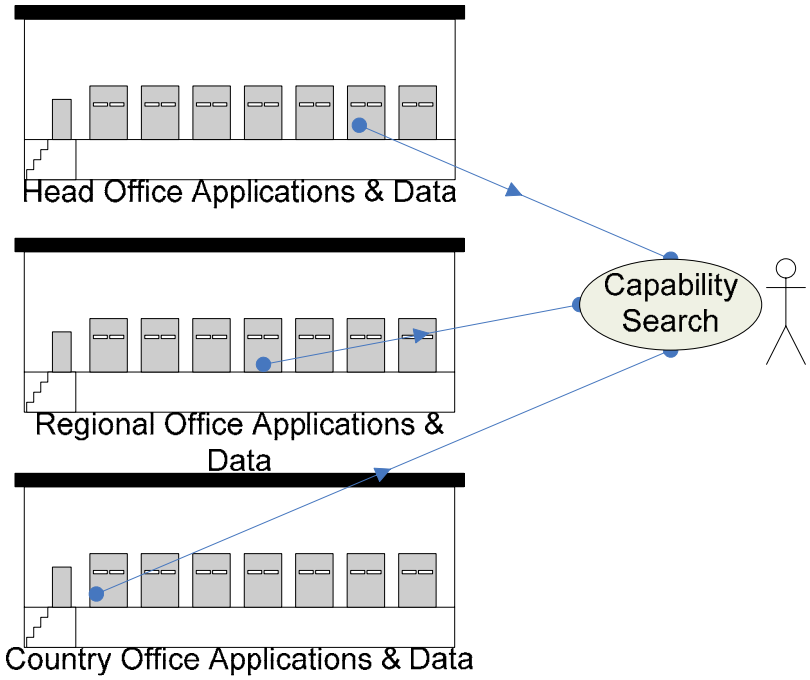


**Figure 2. Business Capability Search Complexity**

Industry currently employs syntactic means of describing these capabilities in repositories such as UDDI (Universal Description Discovery and Integration) [2, 3] and MDS (Monitoring and Discovery Service) [4, 5]. For the research here, both approaches were discounted because of their syntactic nature. As a result, semantic techniques have been chosen because of the need to understand the complex relationships between system capabilities and the domain. With a combined need to understand (a) how service semantics are used and (b) what form they take warrants the practical nature of this research. The development of service ontology and semantic discovery tools provide the mechanics to address this need. Another example of this practicality can be seen in the use of real world user cases to form scenarios. The resulting contribution lies in the framework proposed to architect business knowledge ontology. This framework is grounded in both real-world business software and developmental experimentation. Additionally, the application of grid to a business software reuse situation is a novel application of a much researched area; providing practical metrics for semantic search.

## 3. Related Work

The aim of the literature review is to summarize the current state of the art, though literature that directly addresses the use of ontology to describe and search for business capabilities exposed as grid services is extremely limited. To overcome this limitation, literature from grid based semantic search, web services and semantic web service ontology is examined and synthesized. Each area has a limited focus: (a) Grid semantic search is resource focused, (b) Web services are standards focused (on problems at a granular level) and (c) the semantic web is focused on static content – with a hard semantic focus when applied to services.

Grid technology has been a deployable option in distributed architecture for a number of years, resulting in both academic and commercial software such as Condor and Load Sharing Facility (LSF). Moving away from load balancing to a service programming model has resulted in recent web service standards integration around the Web Services Resource Framework (WSRF) [6]. Instead of executable components being distributed on the Grid, WSRF provides a development model for engineering stateful service-oriented systems. Business grid services are software capabilities resident in operational enterprise applications and are differentiated from the web service in two ways: (1) The service itself is not clearly identified at a fixed endpoint as several service instantiations may

exist for the required service and (2) The stateful grid service programming model extends current web service models with asynchronous messaging that enables state to be observed during operation. For reference, the grid and web service infrastructures are aligned within a common stack (see Figure 3).

The subject of ontology, a term borrowed from philosophy [7], is the study of the categories of things that exist or may exist in some domain [8]. An ontology (in its simplest form) is a catalogue of the types of things that are assumed to exist in a domain of interest from the perspective of a person who uses a language for the purpose of talking about a domain. The types in the ontology represent the predicates, word meanings, or concept and relation types of the language when used to discuss topics in the domain. An uninterpreted logic is ontologically neutral: It imposes no constraints on the subject matter or the way the subject is characterized. Logic alone says nothing about anything, but the combination of logic with an ontology provides a language that can express relationships about the entities in the domain of interest. The use of Ontology within scientific grid projects [9, 10] combines ontology and rules for solving resource matching problems. Although introduced into the grid – with a focus on traditional resources such as hardware and centralized ontology - further work is warranted that addresses its business application with richer services residing in distinct communities. Within a grid context, Blythe et al. [9] have used ontology to make workflow models concrete through the use of a workflow generator. One of their approaches generates workflow using abstract application components, reducing upfront work understanding the inventory in detail. A business grid of more sporadic service deployment and selection is not easily suited to this a priori definition however- particularly when re-use is to be nurtured at the grass roots of the organization. Business application components, unlike scientific grid resources, are unlikely to conform to common semantic models at even an abstract description level. This is due in part to increased organizational size with larger resource numbers (the majority of which are heterogeneous software capabilities). Issues such as semantics, expressiveness, dependency and adequacy, for example, have already been raised in this regard [11].

The World Wide Web Consortium (W3C) Web Services Architecture Working Group defines web services as "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically Web Service Definition Language WSDL). Other systems interact with the web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards” [12].

Essentially, web services can be thought of as "remote procedure calls over the web". The discovery of web services, typically carried out using a UDDI registry, provides a yellow page style lookup on available services. The approach relies on common business and service categorizations having utility across communities in a symmetric manner – difficult across global business organizations such as investment banks. Alternatively, organizing the inventory containing services resident within these systems has brought a resurgence of workflow application, with the Business Process Execution Language for Web Services (BPEL4WS) being the result. Workflow initiatives (such as BPEL4WS, XLANG, WSFL, BTP, WS-C+T, WS-CAF, WS-Choreography) generally focus on representing service compositions where process flow and binding are known upfront [13]. As a point of note, however, discovery provides the focus of the research here, as synthesizing or organizing business grid services is too ambitious at this point.

The semantic web has added ontology to the web services stack; supported through the Resource Description Framework (RDF) and schema (RDFS) languages. These semantic languages have enabled the relation between web resources (including sub-classing) to be made explicit.

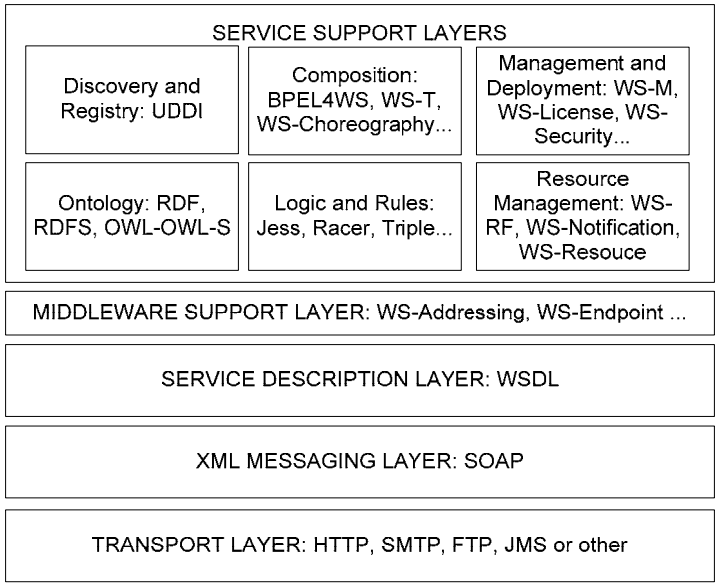


**Figure 3: WS Technologies**

The Semantic Web is rooted in the Scientific American article from Berners-Lee et al. who state "The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in co-operation." [1] The initial ideas were targeted toward static web pages

and it is unsurprising that they gained resonance in the services community. This transition can be seen in the often cited paper by [14] who describe semantic web services as making web services "computer-interpretable, use apparent, and agent-ready". This same technology support cooperation within a business context – providing a use apparent service selection environment.

Current intersections between web services and the semantic web have delivered a diverse body of research. The agent community [14-17] has recognized the benefit of ontology if computer-to-computer web architectures are to be achieved. Furthermore, the combination of service and domain ontology is seen as a key to achieving service synthesis [13]. Work on service ontology is currently centered on the OWL-S and WSMO groups. Recognizing the progress, by the DAML Consortium and others, attention has moved from the ontology languages to specific application to services. A discussion of semantic web services would not be complete without coverage of the OWL-S upper ontology model. The WSMO [18] approach has many similarities with OWL-S, with additional focus on service goals. Within this review, OWL-S provides an adequate exemplar of hard service ontology. The high level model describes the relationship between the differing service decompositions (see Figure 4) [13, 19]. A resource provides a service that is represented by the *ServiceProfile*, described by the *ServiceModel* and supported by the *ServiceGrounding*. Generally, the profile describes the service in a high level way (enough to discover the service), the model describes the detail of how it works and can be used to: (1) perform more in-depth analysis of whether the service meets a need, (2) to compose service descriptions from multiple services to perform a specific task, (3) during enactment, to co-ordinate activities from participants and (4) to monitor execution [17]. The service grounding details practical access and binding.

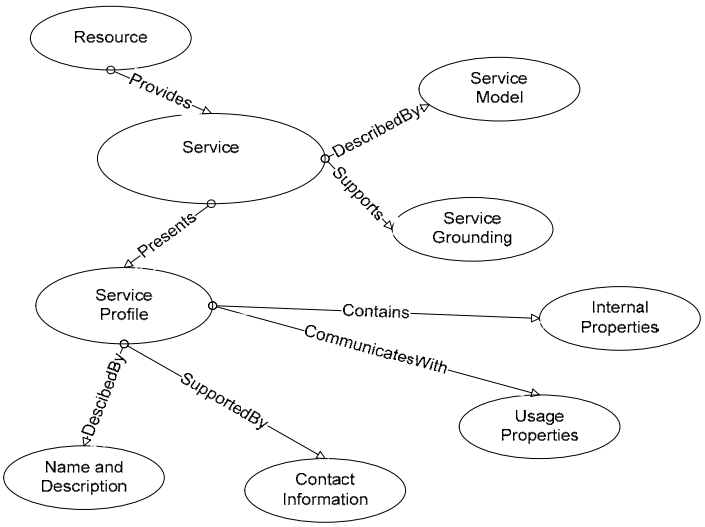


**Figure 4: OWL-S Upper Ontology**

OWL-S delivers a generalized, hard semantic model for describing services. Others have identified the need for specialized common concepts within a web service context [20-26], with one example being quality of service. These concepts represent glue homogenizing a wealth of asymmetrically described services. New issues become pertinent in a semantic web of a "great number of small ontological components consisting largely of pointers to each other" [27 p.31]. Synthesizing the major points of the review, semantic web service communities have proposed hard ontology to describe services and recognize the need to combine service and domain ontology, and the grid community has focused on hardware resources. The operational nature of software grid resources require service knowledge be mined from the operational systems. Undertaking this with centralized, hard ontological structures will not address the distributed ontology concerns raised by Hendler [27]. Consequently, there is a need to address the application of the semantic grid to business grid services. An important early endeavor is to construct service ontology that represent such business grid services, and include temporal characteristics.

## 4. Research Design

The foremost question in semantic service orientation is how discovery and synthesis should be undertaken most effectively. The research described in this paper initiates this process by scoping the use of business grid services and associated ontological service knowledge. The research follows a design research approach, which is a "search process to discover an effective solution to a problem" [28 p.88]. The relevance of the problem for the

research community must be demonstrated and the solution must be effective to a satisfactory level. The effective solution may not (and generally does not) coincide with the "best" or "optimal" solution however - generally the effectiveness of the solution must be demonstrable through an iterative evaluation of the designed artifact(s).

The design research presented in this paper is methodologically based on and adapted from the approach described by Nunamaker et al. [29] and the guidelines presented by Hevner et al.[28]. The research outputs are also be described according to March and Smith's [30] terminology for design research.

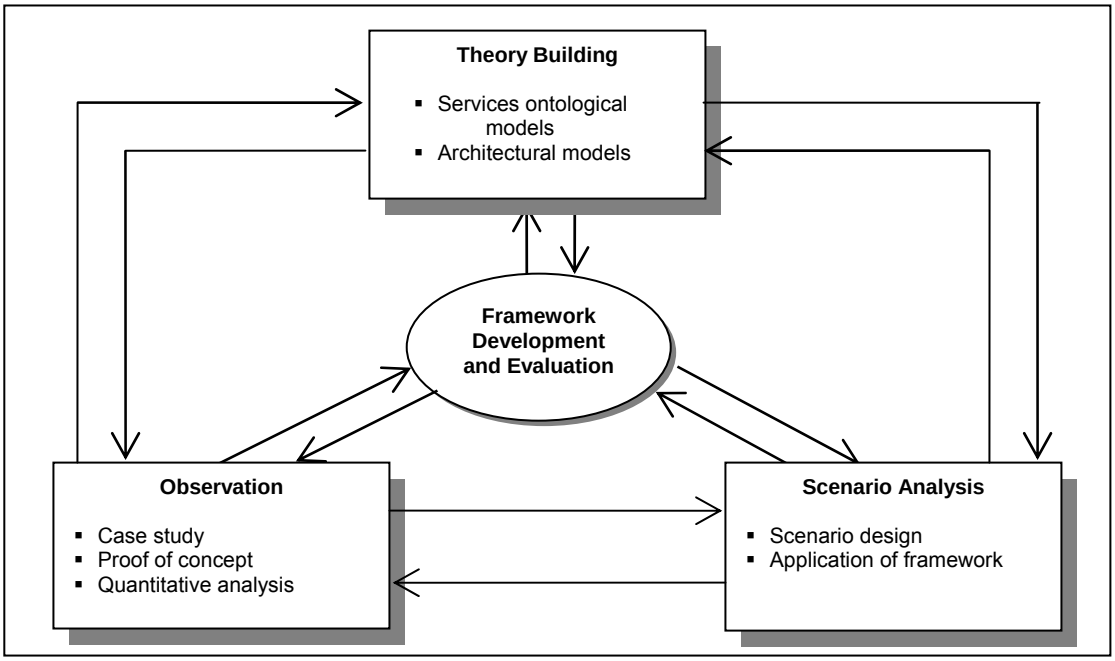


**Figure 5: Framework development and research strategies (adapted from Nunamaker et al.** [29]**)**

Adapting Nunamaker's multimethodological approach to design research [29] the following four research strategies are adopted (Figure 5):

- Theory building: The study is theoretically based on previous work conducted in the areas of grid based semantic search development and ontological modeling. The proposed system and framework builds upon this theory by covering an existing gap represented by its application to a new form of grid service – the Business Grid Service. Although the dominant literature concurs on the necessity of ontologically grounding service descriptions, limited work has been carried out on the convergence of service models and ontologies of this type. More specifically the research builds on the previous work on interpretation of

system models to derive ontological models and semantic search. Theory is derived from the design of a discovery system that uses the aforementioned ontological models. Design theories are further uncovered whilst these ontological models are implemented and analyzed.

- Scenario analysis: The developed framework has been evaluated primarily through its application to typical web services re-use scenarios. The scenarios are recognized capability requests drawn from experience and observation in this type of live industrial project. The application of the framework to the scenario represents the development of a "proof of concept" discovery project whose outcomes are to be evaluated.
- Observation: Four sets of observations were conducted. The first concerned a study of a three live industrial projects in which similar software capabilities were deployed. This study was undertaken in order to understand, within the financial services (FS) domain, how typical software is modularized and described. This enabled the initial service ontology to be realized. Further observation was carried out on the implemented discovery architecture in order to evaluate the framework and the framework derived ontology.
- Framework and ontology development and evaluation: Design iterations are evaluated quantitatively and in relation to business user case scenarios. The service ontology provides tangible artifacts for evaluation. Variation is business grounded ontology directs much of this – during development and experimentation. The framework is formed through reaction to different ontological models whilst undertaking the research process.

The above strategies permeated the research as a whole. The strategies themselves should not to be considered as process steps, but rather as means of organizing the researchers' thought processes. All strategies were influential during every step of the study. In terms of the iterative cycle adopted to materialize the various research artifacts (i.e., constructs, models, method and instantiations), the steps that were followed are schematically outlined in Table 1. It is the combination of industrial observation, implementation of theoretical artifacts and real-world scenarios that combine to enable interpretation of the framework content in situ. The initial ontology and discovery architecture are evaluated quantitatively and result in iterative development of the framework. The framework as a whole is evaluated at the end of the improvement and quantitative evaluation cycle. Refinement is concluded when

the discovery architecture and ontology combine to effectively support the scenarios in the context of global FS organizations such as Investment Banks.

| Phases of research | Individual steps |
|---|---|
| Identify problem relevance | Conduct literature review<br>Analyze industrial case study<br>Identify gap(s) |
| Framework design | Define scope of framework and architecture<br>Define underlying concepts and constructs<br>Define framework process<br>Define framework artifacts (input and output) |
| Framework evaluation | Apply framework to a realistic scenarios<br>Observe framework in action with proof of concept |
| Improve and re-evaluate framework | Identify limitations or areas of improvement<br>Refine (re-design) framework (iterate previous two steps) |
| Communicate and discuss research | Identify limitations and further potential benefits<br>Define directions for future work<br>Disseminate (e.g., present and publish findings) |

**Table 1: The adopted design research process (based on guidelines by Hevner et al.** [28]**)**

The previous section observed a gap in the synthesis of relevant literature. The following sections will provide further evidence of such a gap, confirming the relevance of the problem investigated. Process steps and outputs are documented in Table 1 and will be presented in a systematic manner. This design research project generates outputs categorized by March and Smith [30] as: *constructs*, *models*, *methods*, and *instantiations*. The organization of these design artifacts are conceptualized within the inner and outer environments, discussed by Simon [31], as a means to position the research activity effectively at the interface between the two environments. The outer environment being the explicit service knowledge in ontological form and the inner environment is physical discovery architecture. At the interface the dual impact of semantic search techniques and service knowledge are uncovered. Models within design research are problems and solutions statements [30].

## 5. Discovery Architecture and Implementation

In order to fully design and examine a Business Grid of software services, grid services must be extracted from realistic use cases. Grounding initial design artifacts in such a use cases allow for:

- The demonstration of applicability to a live, large-scale software system. In particular, the way in which software is modularized and as such how such software is syntactically described. In other words, the syntactic content provides a usable basis for semantic modeling.
- To make the case for a framework that extracts the semantics from syntactic capability description and operational characteristics ready to transform that content into rich technology-agnostic representations.
- To design a realistic scenario to evaluate and refine the framework defined by this research. Due to the current level of immaturity of the Semantic Grid/Web and the skepticism of some (including both researchers and practitioners), the framework could only be experimented on a simulated pilot-project. Consequently, scenarios are designed that utilizes experience and domain expertise gained from previously developed commercial business services.

Exploring the use of grid service technology in Financial Markets organisations can only be carried out through understanding their service offerings. Recognising this, the research has brought together software capabilities from three projects within a single organisation. HIBANK is an Investment Bank providing derivatives and fixed income trading operations to corporate clients from which three systems from three distinct locations were analysed. For reference, these systems combine thick clients in Java, thin web clients, Excel clients using .NET framework and B2B clients. HIBANK is undertaking a strategy to integrate corporate systems and processes – citing duplication as one of its concerns. Deeper investigation into similar software artifacts reveal specializations that have led to this so called duplication. More specifically, examples of the functionalities found in the systems analyzed (and common amongst the different systems) include:

- 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 trade types
- 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)

Technology architectures, mimicking the business they serve, are often segmented by product, process or geographic concerns. Two examples of development are moving into a new market or developing a new business product from existing parts. Groups of capability within a segment are often conceptualized and described using their own specific language and terminology. The specifics may only be subtle, but result in difficult mapping to a new market or ambiguity describing a new product. These differences were clearly present across the three systems (OptionPricer and BlackScholes pricing trades in different system using exactly the same algorithm for example). Relevancy of the research is clearly seen in this example.

The lack of semantic support in a business capability re-use context is clear from the stated examples. The methods are process steps undertaken to perform the task – the framework for architecture discovery on a business grid. To summarize the design, a prototype instantiation is developed in order to investigate the theory that novel business grid service architecture is best supported by semantic models whose content is combined from artefact description and operational characteristics. The semantic models themselves are developed to support semantic integration, and include the fit for purpose and levels of complexity resident in the systems in which they are grounded. Consequently, much of the evaluation is performance or process related. Grid and web services evaluation is carried out using three real-word scenarios:

- *Use-case 1 – Searching for trades executed with a particular counterparty.*
- *Use-case 2 – Valuing a portfolio of interest rate derivative products.*
- *Use-case 3 – Valuing an option based product.*

These three core use-cases were chosen because they provide examples of three distinct patterns of use. Use-case 1 requires data access to several systems, *aggregating* results into a composite portfolio. Use-case 2 uses a relatively *standard*, unique capability, whereas Use-case 3 may use one of several *alternative* capabilities. Methodological modeling of state, for improved selection accuracy, directed theory and subsequent implementation of ontology specialization methods. The impact of a general state integration technique required analysis; and 3 further use cases were used to simulate business state growth. These scenarios cover the caching in stateful services, either as a lists of trades (books) or discrete trades:

- *Use-case A – Risk Management where 10 books each hour are cached in the underlying grid service.*

- *Use-case B – Pricing a Low Volume Product where 40 new trades per hour are cached in the underlying grid service.*
- *Use-case C – Pricing a High Volume Product where 200 new trades per hour are cached in the underlying grid service.*

### 5.1. Financial Service Capability – Where and How

The geographical coverage of an Investment Bank motivates a design that enables both discovery architecture components and service ontology to be placed across the network. The knowledge base, one or more OWL files, is easily hosted and replicated on one or more hosts. The matching algorithm can be deployed on one of four tiers; the client, a front line hosting server, a grid node or with the knowledge base. The client applications, for the purposes of this research, comprise a classic Java client, using JAX-RPC to use the discovery service, or a Microsoft .NET client. A further alternative can place a thin client on the server side or a declarative client generated on the server side but rendered on the client side. The flexibility is purposefully restrained for testing, using the following setup:

- All web ontology OWL [32] based knowledge files were placed on a single host.
- The client application options allowed operation as a client-server declarative client or a thin HTML client.
- A grid service tier containing the business service capabilities.
- Middleware services resided on a single host.

#### 5.1.1. Discovery Architecture

The Semantic Discovery for Grid Services architecture (SEDI4G) (depicted in Figure 6) was iteratively developed and evaluated (directing the framework construction). Development evolved from a single client-side application to the distributed form shown. The separation of knowledge is realistic [27] as it provides a means to integrate service ontology or higher level domain taxonomies.

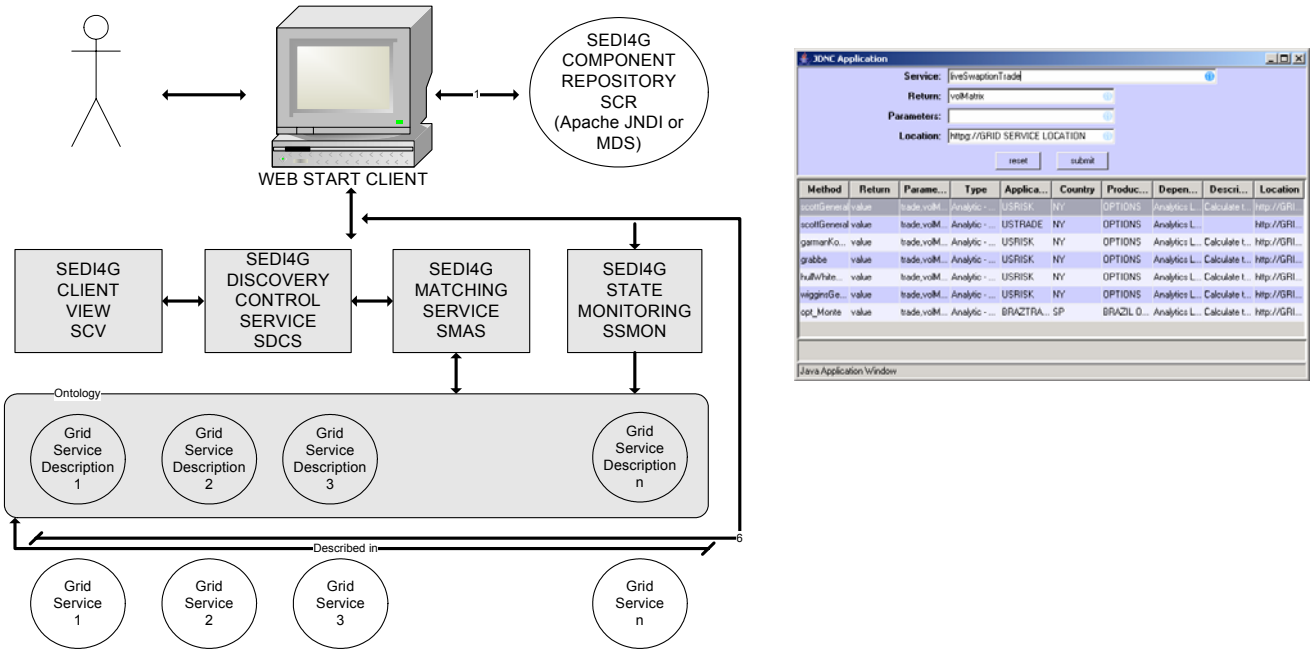


**Figure 6. SEDI4G Discovery Architecture**

The distribution of components on the network is also influenced by the changing network economics highlighted by Gray [33]. This version of the SEDI4G architecture thus enables multiple ontologies to be used independently, supporting the mix of knowledge requirements and differing sources (human and machine). Several business applications from the FS domain were mined for capabilities (e.g. Object methods in a C++, .NET or Java based systems). These capabilities were then (1) stubbed as grid and web services and (2) added as classes, with parameters as properties, to the service ontology. Duplicate software components that value Interest Rate Swap and Interest Rate Option products were resident in both trading and risk management systems. The components descriptions are brought together within the ontology. The resulting service sets cover trade execution, price calculation, risk management, settlement and credit risk. The use of *de facto* business terminology and object generalization required that certain terms be further analyzed. An example is the use of the term “trade” to describe the various states of the trade – planned, new, live and dead. These specializations were recognized and made explicit manually in the first iteration OWL ontology. Subsequently, recognizing the importance of this specialization motivated the development of automatic state integration into the OWL ontology through capability observation.

### 5.1.2. Semantic Search Process

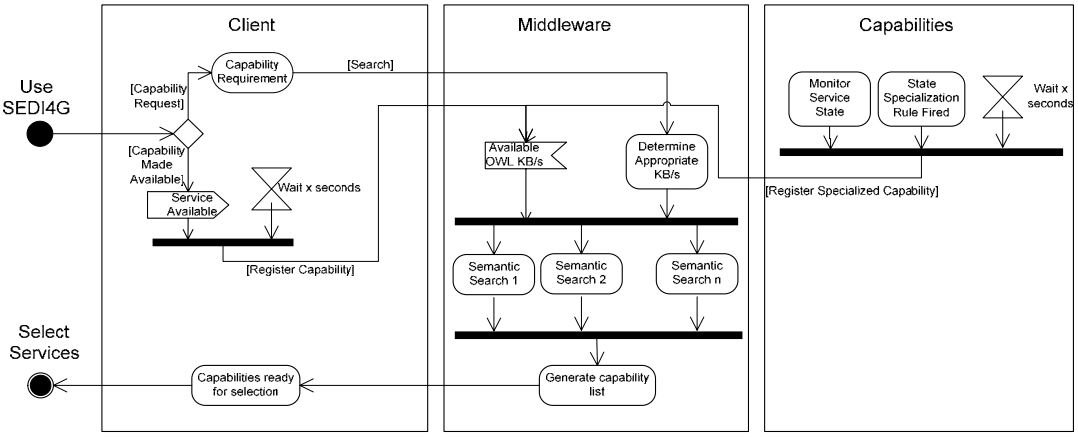


**Figure 7. Semantic Discovery Activities**

The discovery process begins by identifying which discovery infrastructure components are required to carry out a semantic search - such middleware components support placement flexibility (represented by the grey flexible services and data in Figure 6). Put simply the system loads service and/or domain knowledge, made explicit in OWL, and queries classes and properties that make up such knowledge. In terms of use, the user specifies search parameters and receives a list of appropriate Business Grid Services. Extension of the OWL files, with specialized classes, result from specialization rules. Examples of such rules involve timing specific resource properties or identifying properties of particular business value. Messaging infrastructure delivers the resource properties for monitoring of this kind. Two such properties within the FS domain were *Books*, a grouping of trades and *Curves*, a grouping of market data. New subclasses are added to the ontology describing newly identified stateful resources. The business grid services simulate the capabilities that exist within the business systems. The activity diagram (Figure 7) exposes the human and machine knowledge processing that results in service classes being recorded within the OWL files.

In an industrial application these services would reside within the business system. Here we have replicated the method signatures and state transition (exposed as resource properties) as grid services; adding probabilistic execution time and success/failure outcomes. For reference, the grid services were implemented in MSOGSI.net, WSRF::Lite and Globus 4 in a *restful* style with a common execution and monitoring semantics. As stated, access to

system or resource properties is abstracted and generalized within the SDCS service. This initial prototype uses a request-reply interface (it is recognized that an asynchronous approach based on WS-Notification is a likely next step).

### 5.1.3. Description of Matching Algorithm

The matching algorithm is based on a semantic method using ontology and a reasoning engine, the background to the subject is introduced in section 3. Much research has focused on single, centralized ontologies – methodologically engineered or systematically integrated. Our approach removes this restriction in order to test the use of distributed knowledge at particular points in the discovery process.

```
KB: Knowledge Base
RETE: RETE Object
URL: URL of ontology
RS: returned services
MS: matched services

KBInitialisation:
RETE ← create_Rete_Object()
read_Jess_Rules_And_Facts()
run_Rete_Inferred_Rules_By_Rules_And_Facts()
KB ← load_OWL_Resource(URL)
runRete_Apply_Semantics_To_OWL_File()

SearchRequest:
for all searchWords do
   RS ← search_Ontology_For_Service(URL)
end for
   MS ← remove_Ambiguous_Services(RS)
return MS
```

**Figure 8. Pseudo Code of Matching Algorithm**

The matching algorithm comprises two steps; the initialization of the knowledge base and the actual search request. During the initialization phase the ontology is loaded and the facts and rules are applied using the Rete algorithm [34]. During the search request the ontology is reasoned depending on the specified Jess queries and the matched services are returned. Figure 8 summarizes the steps carried out by the matching algoritm.

## 5.2. Discovery Architecture and Implementation

The prototype system was implemented using Protégé (with OWL Plugin) and used to build three service ontology OWL files (small, medium and large). The discovery algorithm is provided as a grid enabled web service. The control web service (Java) directs control to the discovery algorithm, which then generates either (a) XML for return to the client, (b) HTML for thin client usage and (c) a JDNC data file. All three clients were implemented – a Java application client, a declarative client using Java Network Launch Protocol (JNLP) with Java Desktop Network Components (JDNC) (see Figure 7) and Microsoft Excel front-end (using the .NET SDK). The flexible placement strategy allows the web services and knowledge base to be discovered using more traditional registry methods; providing a discovery system for the semantic searching components.

The lack of common service discovery systems demonstrated a lack of expressive service description and the lack of infrastructures for defining common service descriptions across enterprises and customers [35]. The lack of expressiveness is further restricted by the generalization of concepts used to describe capability. To provide an enhanced discovery for the FSs, the prototype is supported by a matching mechanism which allows for a more efficient service discovery by using a FS ontology described in a semantic language and a reasoning engine that uses the ontology [36]. The flexibility in capability categorization and placement address many of the limitations exhibited by fixed information models such as OWL-S and centralized repositories such as UDDI.

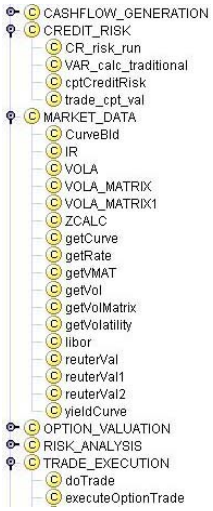


**Figure 9. FS Ontology Structure**

OWLJessKB [37] is used as the basis for the capability discovery process: It is intended to facilitate reading OWL files, interpreting the information as per OWL and RDF languages and allowing the user to query on that information. The service discovery works in the following way. Firstly, the tool inserts the triples as 'facts' into the Jess knowledge base [38]. Then, with predefined rules, Jess can reason about the triples and can draw more inferences. The Jess API (Application Programming Interface) is intended to facilitate interpretation of information contained in OWL files, and it allows users to query on that information. The Jess API leverages the existing RDF API to read in the OWL file as a collection of RDF triples. Figure 9 shows a part of the FS Ontology. The services are listed as classes of the ontology, having methods, which are sub-classes, and parameters which are properties spoken in ontology terms. The user defines, for example, three search words which are: `executeTrade`, `boolean` and `trade`. For all three search words the Jess query shown in Figure 10 is invoked. The organization of classes within the ontology relies on a level of expert knowledge; understanding the capability in relation to the rest of the inventory and its underlying properties. The inputs and outputs of the capability often direct organization within the ontology. This methodological approach is in contrast to the specific matching of input and output properties [21] – where those properties take a more significant role. The combination of all properties, further enhanced with state knowledge, provides a more descriptive object from which to select available resources. Input and output focus within the ontology is relaxed because it is assumed that the ability to transform existing instances will already reside in many businesses, and can be seen in the many commercial connector frameworks (e.g. J2EE Connectors or IBM MQ).

For each returned class, the Jess query shown in Figure 11 is invoked. This returns the services `getTradeByID`, `getTradeByDate`, `getTradeByType` etc. The sub-classing query finds another three related services, which is basically achieved with the subsumption relation, which defines a hierarchy of concepts. Finally, the services are returned for display and user orchestration.

```
(defquery query-for-class-of-a-given-property
"Find the class to a given property."

  (declare (variables ?class))

  (triple
    (predicate "http://www.w3.org/2000/01/rdf-schema#domain")
    (subject ?class)
```

```
    (object ?x)
  )
)
```

**Figure 10. Jess Query – Property of Class**

```
(defquery query-sub-class
"Find all the sub-classes."

  (declare (variables ?y))

  (triple
    (predicate "http://www.w3.org/2000/01/rdf-schema#subClassOf")
    (subject ?x)
    (object ?y)
  )
)
```

**Figure 11. Jess Query – Sub-Classing**

## 5.3. Performance Measurements

In order to fully explore the practicalities of ontology size and placement – and contrast to that of centralized ontology – the business use cases are executed against a set of business grounded test ontology. The placement of ontology files require exploration – with ontology being placed on local and distributed hosts. The vertical structure of the ontology is also examined. Finally, the input of specialized operational knowledge – examined using specific growth use cases – is tested. The performance testing is carried out within the overall design framework, where knowledge and architectural artifacts are considered and refined. Performance measurements were carried out using the SEDI4G distributed discovery architecture presented, utilizing various topologies and ontology sizes in order to determine size constraints imposed by using this richer service ontology.

### 5.3.1 Grid Service Search

The prototype system uses three core business-grounded ontologies (summarized in Table 2) to investigate the impact of KB size on semantic search performance. The ontology size is dictated by increases in the number of product, process or geographical systems being described.

| | Small Ontology | Medium Ontology | Large Ontology |
|---|---|---|---|
| Number of Classes | 57 | 114 | 224 |
| Number of Properties | 23 | 46 | 92 |

**Table 2. Phase 1 Ontology Comparison**

Results shown in Figure 12, timed milliseconds, cover server time, both in instantiating the OWLJessKB objects and executing the semantic search. Client side and initialization figures highlight the need for server side caching as initial runs reveal an overhead in the region of 10 seconds. The timing labels are new object initialization (N), finding semantic matches (M) and the total (T), which are taken from services that are already initialized. As seen from the measurements, the longest timings occur during the matchmaking process, due mainly to the use of the Rete algorithm. This algorithm is intended to improve the speed of forward-chained rule systems by limiting the effort required to recompute the conflict set after a rule is fired - its drawback is that it has high memory space requirements due to the loading of the ontology into the Rete object. Once created and set, the algoritm it is fast to call rules and queries in order to infer the semantic relations of the ontology loaded. The results highlight the near linear degradation in performance exhibited by the search. In a dynamic system context, ontologies larger than around 500 classes become impractical. Extrapolating the results to reflect the investment banking domain – where thousands of granular capabilities exist - a single, centralized ontology will not provide on-demand performance.

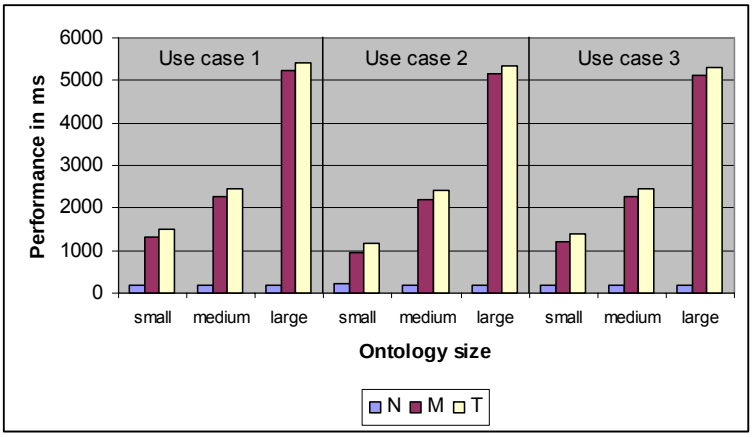


**Figure 12. Ontology Size Results**

A second set of measurements investigated the impact of a local versus a distributed setting of the SEDI4G knowledge - with the ontology files placed on a remote machine.

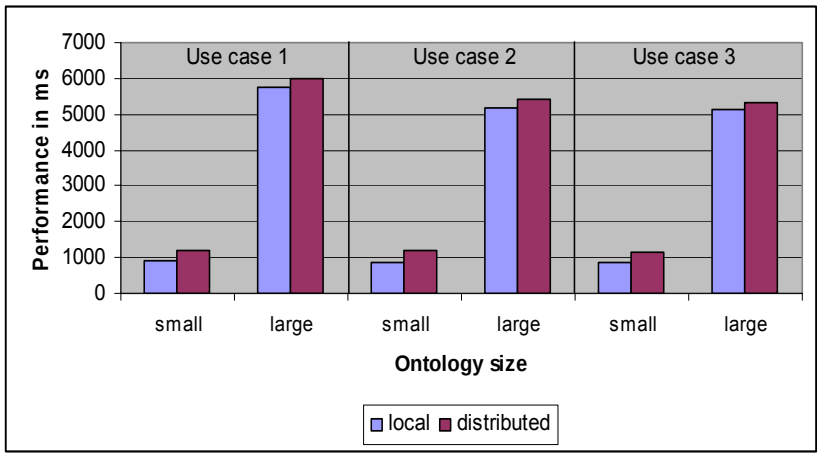


**Figure 13. Performance Test - Local vs. Distributed**

Figure 13 shows the performance measurements regarding the three use-cases and two ontology sizes placed locally or remotely on the wide area network. Measurements taken in this experiment focus on the transfer time of the ontology file from the hosting site to the search component. Comparing the results quantitatively reveals that the communication overhead, the time taken for the file to be transferred, has a greater impact on the small ontology than on the large ontology. Overhead were 24%, 28% and 24% for the small ontology and 5%, 4% and 3% for the large ontology. Obviously, this is due to the fact that the relative time taken loading the small ontology and the application of the rules were much smaller than for the larger ontology, and should be considered alongside ontology size.

| Ontology Size | Performance with locally situated Ontology (ms) | Performance with distributed Ontology (ms) |
|---|---|---|
| Small | 883 | 1182 |
| Large | 5362 | 5576 |

**Table 3. Average Results of Local vs. Distributed Performance**

Table 3 summarizes the average performance values for the local versus distributed setup. The implications of this communication overhead are that on a good (broadband or better) network the relative overhead is small (4%) when searching several hundred classes. Conversely, a small, distributed KB result in a 25% performance degradation and should dictate a topology that brings such artifacts close to the search space.

### 5.3.2 Vertical Categorization

So far, the evaluation has covered the replication of services knowledge horizontally within a relatively balanced ontology. Obviously, the structure of business service semantics cannot be assumed (clearly apparent in the differences found in published domain ontology and the observed business software components), but still warrant some exploration. Considering the vertical component of the ontology structure, an additional experiment is carried out by extending the current service ontology from three to seven levels (seen in Figure 14).

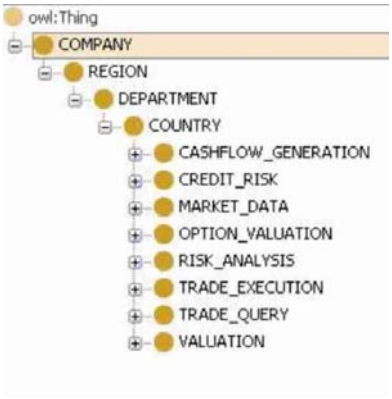


**Figure 14. Additional Categorizations**

Timing the semantic search (including fact generation and search) has provided some interesting results:

| Ontology Depth | Number of Classes | Average Search Performance (ms) | Performance Degradation (%) |
|---|---|---|---|
| 3 | 57 | 1196 | |
| 4 | 58 | 1270 | 8.3% |
| 5 | 59 | 1313 | 5.7% |
| 6 | 60 | 1386 | 7.0% |
| 7 | 70 | 1432 | 5.2% |

**Table 4. Ontology Depth**

Performance degrades by between 5.2% and 8.3% with no recognizable trend. These results direct the design of OWL based ontology away from unduly deep taxonomies. This prognosis sits well with service ontology (such as OWL-S or WSMO) with service characteristics described with limited depth. In the case of the experimental data, shallow service type ontology delivers better performance.

### 5.3.3 Evolutionary Ontology

Recognizing the value of business grid services that are in a particular state – requires that specialized sub-class definitions are identified and made explicit. The heuristics needed to determine the benefit of specialized service description are difficult to generalize within the business domain, but the impact of such heuristics warrant analysis. The second testing phase simulates three scenarios where the ontology are extended though autonomic service monitoring – termed dynamic ontology growth scenarios (Table 5). The resulting ontologies were compared using the same three use case searches (see 5.1). The results are shown in Figures 15. The identification of state properties that have a particular value needs exposition.

| | Growth in Classes |
|---|---|
| State Growth A | 80 |
| State Growth B | 320 |
| State Growth C | 1600 |

**Table 5. Ontology Growth**

The addition of specialized sub-classes to the ontology provides greater real-world precision, but at a cost. It can be seen that, even when taking this small slice of capability, the ontology growth makes it near impractical to use a consolidated knowledge base. The consequence could be to discard the approach, which would be a mistake if the relative merit of this specialized capability outweighs the query performance.

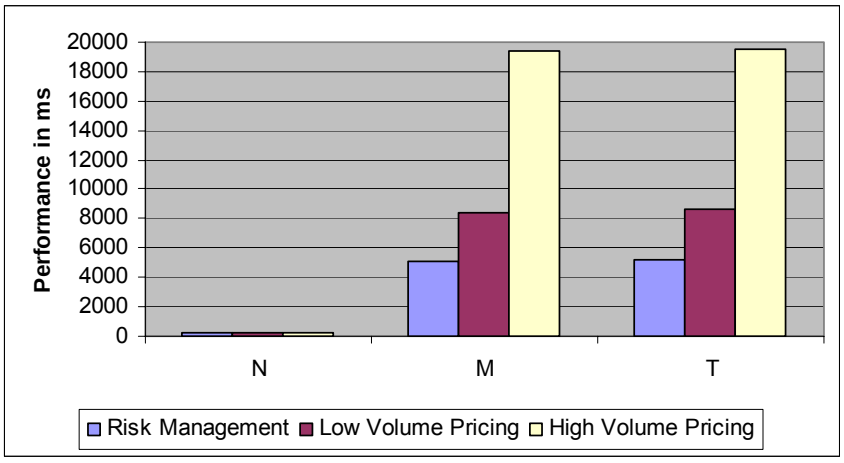

**Figure 15. Combined Growth Results**

One alternative is to separate the state service knowledge from the combined KB: It is, after all, already supported by the pre-discovery selection phase of the SEDI4G architectural components. The critical question is that of how the service knowledge is separated. Here we have provided a basic approach to this by federating the state knowledge bases, which are then selected in pre-discovery when a query identifier specifies a particular KB. The linear degradation in query performance does highlight that small service ontologies with minimal change in state are capable of remaining in a centralized form. The measurements taken provide the means to provide a regression line that can be used to describe such performance traits. The use of analytical approaches cannot be generalized as different source systems are likely to yield different results.

## 6. Business Grid Ontology Framework

The performance results, in Section 5, provide a quantitative exploration of three characteristics that affect the design: (1) the size of the ontology, (2) the placement of the ontology and (3) the addition of operational business state as ontology sub-classes. Although quantitatively important in directing OWL file implementation, it is the processes at the interface of discovery and knowledge that generate the theoretical framework (see Table 6).

| Activities | Description | Input Artifacts | Output Artifacts |
|---|---|---|---|
| Business Grid Service Identification | An application component is evaluated for inclusion in the Business Grid. The component interface is interpreted. Available state signs are modeled as WSRF resource properties. | ▪ Component Interface, code, documentation<br>▪ Classes are identified and related to the domain | ▪ Business Grid Service Description in WSDL<br>▪ Draft Ontological Classes |
| Business Grid Service Categorization | Business Grid Service descriptions are described using the interpreted classes. | ▪ Grid service descriptions (e.g., WSDL code)<br>▪ Classes | ▪ Draft Service Ontology |
| Business Grid Service Property Evaluation | The concepts represented in the service models are either mapped to pre-existing domain ontologies or assigned to newly developed ones. Properties that have value are identified and stated in rule form. | ▪ Draft Service Ontology<br>▪ Domain ontologies | ▪ Objects incorporated or mapped to ontological domain models<br>▪ Property value statements |
| Business Grid Service | Ontology models are created in an ontology language such as OWL. The ontology is deployed and | ▪ Service Ontology<br>▪ Domain ontologies<br>▪ Property value | ▪ Service Ontology OWL Files<br>▪ Ontology |

| | | | |
|---|---|---|---|
| Ontology Realization | resisted with the monitoring services. | statements | monitoring rules |

**Table 6. SEDI4G Architecture Framework**

The SEDI4G Architecture Framework supports the implementation of business grids – from capability identification through ontology creation. The framework steps required to implement such a system comprise:

- Business Grid Service Identification. Capabilities that exist within selected enterprise systems are analyzed and capability names and parameters are extracted. The description is formalized into a WSDL document ready for use in developing or re-factoring the capability as a business grid service. WSDL, although syntactic in nature, allows the analyst to map properties onto common domain properties available in WSDL. These domain properties may range from simple WSDL types such as String to the use of complex domain Schema. Syntactic modeling at this stage enables the analyst to identify likely ontological classes that will be required in the Service ontology.
- Business Grid Service Categorization. A first service ontology is created, with capabilities becoming classes, methods becoming sub-classes and parameters becoming properties. The OWL file generated may form an OWL-S, or WSMO, service description; or a separate service type ontology utilized by one of these harder semantic models. Domain categorization can be applied, as super classing.
- Business Grid Service Property Evaluation. The content of the first service ontology requires re-evaluation on two fronts. Initially, structural changes may be required with respect to the performance metrics. The second evaluation is carried out in order to identify properties of specific business value. These properties are added to the ontology for use by the ontology evolution algorithm.
- Business Grid Service Ontology Realization. The finalized ontology files are generated. Harmonization between service ontology, domain ontology and rule base is carried out for practical deployment issues. The issue already identified, such as ontology placement, size and depth, play a major role in this task.

Revisiting the performance results of most significance relates to the size of the ontology. The metric highlights the performance boundaries associated with larger ontologies. This finding raises questions about the need to integrate service knowledge for the sole purpose of better selection. Within an environment of several thousand classes, easily seen in financial services or with large amounts of state inclusion, a federation of ontologies

across the network is valid – using a hybrid ontological pattern [39] with shared vocabularies. This is supported by the network performance seen in local and remote ontology testing (with the premise of good network bandwidth). Reducing the amount of the ontology model being used may be appropriate in some cases. This approach has some similarity to pruning [2], as it limits the numbers of facts being processed, and more importantly no requirement to integrate ontology.

## 7. Evaluation of Artifacts

Evaluation of the research is directed by Simon's [29] proposition that design science is carried out at the interface between inner and outer environments. The interface in this study is between the development of ontological models and their utilization when discovering services. Two artifacts represent these environments:

- Business Grid Ontology Framework. The framework encapsulates the methods used to create ontological service models that improve the effectiveness of service discovery in the FS domain.
- Discovery Performance Measures. The measurements validate the practicality of such models to be applied to the domain of study. The effectiveness of differing architectural and topological models of discovery components and KBs are included in these measures.

Adopting a design research approach to this area, with its support for process and constructed artifacts, has proved to be an auspicious choice and enabled the development, validation and refinement of the framework. The holistic nature of the design approach has enabled practical issues to be resolved within the research iterations. Examples of such issues being: (a) the ability to support distributed ontology, (b) limiting ontology depth and (c) extending ontology both manually and automatically for added precision. The use of the OWL-S profile to bind together service and domain ontologies provides a root for standard semantic searching by traveling throughout concept branches of particular domain ontological models. The OWL-S profile will explicitly define the domain ontology being used. Consequently, this will allow the distributed search to deploy the semantic search service to an optimum place on the network with respect to the domain ontologies. Important if such files are small and many (as the small, distributed ontology overhead was 25%). Contrasting with a UDDI fixed syntactic hierarchy, the semantic search over several service models grounded in real-world "things" provide a greater scope for matching to a requestor's concept. The framework provided valuable guidance in that it embodies practical design theories.

Quantitative measures are evaluated in section 5 and validate the effectiveness of the models when used for the semantic search scenarios. The number of services returned, when executing each scenario, is higher than can be achieved by syntactic methods. This outcome is a natural consequence of semantically modeling similar services. Extending the model to evolve under operation provides opportunities for (a) more precise service choice and (2) inclusion of many operational services instead of a single abstract service.

## 8. Conclusion

The work here has reported on a framework to enable the practical convergence of semantic web services and grid is proposed to support the reuse of enterprise system components. Critical to this approach is the use of operational business state to select and utilize business grid services. The approach has followed a design research methodology and combined commercial application analysis, semantic search development and scenario execution to direct an architectural framework. Theoretical development of the framework and search tools arose out of financial services systems, ontological models and development, and subsequently directed further iterations of the design.

Development of the SEDI4G system and performance results show that dynamic re-use of service capabilities is easily and flexibly supported by the business oriented discovery service. The performance of the semantic search (when fully optimized) is acceptable for use within this domain for ontologies up to 200-300 classes (the large ontology test case). The performance degrades linearly with vertical ontology extension and between 5.2% and 8.3% with horizontal as classes are added. Likely centralized ontology sizes within an Investment Bank extend into thousands of classes and justify the separation and use of distributed knowledge pools. Distributed service ontology support is a key characteristic of the SEDI4G architecture, allowing ontologies to exist within its search scope (a prerequisite when thousands of classes exist across the organization).

The use of resource state heuristics to dynamically extend the service ontology with more precise classes representing enterprise system state is novel and provides one contribution of the work. The application of semantic search within this context provides a second contribution: Again this application is novel in approach and supports the semantic integration required to enable the resulting heterogeneous business capability inventory. The complexity of the business domain makes generalization difficult outside the financial services area under analysis.

The SEDI4G Architectural framework provides a contribution in the form of a theoretical engineering framework for applying semantic search and business grid service ontology to this field. The application of the framework, either academically or within industry, to additional systems and domains will make such generalization possible.

## 8. References


[1] T. Berners-Lee, J. Hendler and O. Lassila, "The Semantic Web," *Sci.Am.*, vol. 284, pp. 34-43, 2001. 2001.

[2] M. Paolucci, T. Kawamura, T.R. Payne and K. Sycara, "Importing the semantic web in UDDI," in *Web Services, E-Business, and the Semantic Web*, vol. 2512; 2512, Berlin: SPRINGER-VERLAG BERLIN, 2002, pp. 225-236.

[3] UDDI.org, "UDDI Technical White Paper", "http://uddi.org/pubs/uddi-tech-wp.pdf"

[4] S. Fitzgerald, I. Foster, C. Kesselman, W. von Laszewski and S. Tuecke, "A Directory Service for Configuring High-Performance Distributed Compuations," in 6th IEEE International Symposium on High Performance Distributed Computing (HPDC-6), 1997.

[5] I. Foster, C. Kesselman, J. Nick and S. Tuecke, "The Physiology of the Grid: An Open Grid Service Architecture for Distributed Systems Integration," 2002.

[6] WSRF, "Web Service Resource Framwork (WSRF) http://www.globus.org/wsrf/," 2004.

[7] T.R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, pp. 199-220, 1993.

[8] J.F. Sowa, "Ontology, Metadata, and semiotics," *Springer Lecture Series*, vol. Conceptual Structures: Logical, Linguistic, and Computational IssuesLNAI 1867, pp. 55-81, 2000.

[9] J. Blythe, E. Deelman, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, A. Lazzarini, A. Arbree, R. Cavanaugh and S. Koranda, "Mapping Abstract Complex Workflows onto Grid Environments," *Journal of Grid Computing*, vol. 1, pp. 9-23, 2003.

[10] H. Tangmunarunkit, S. Decker and C. Kesselman, "Ontology-based Resource Matching," in 2nd International Semantic Web Conference (ISWC2003), 2003.

[11] S. Agarwal, S. Handschuh and S. Staab, "Surfing the service web," in *Semantic Web - Iswc 2003*, vol. 2870; 2870, Berlin: SPRINGER-VERLAG BERLIN, 2003, pp. 211-226.

[12] A. Brown and H. Haas, "Web Services Glossary - Web Services Architecture Working Group," 2005.

[13] L. Chen, N.R. Shadbolt, C. Goble, F. Tao, S.J. Cox, C. Puleston and P. Smart, "Towards a Knowledge-based Approach to Semantic Service Composition," in Second International Semantic Web Conference (ISWC2003), 2003.

[14] S.A. McIlraith, T.C. Son and H.L. Zeng, "Semantic Web services," *IEEE INTELLIGENT SYSTEMS & THEIR APPLICATIONS*, vol. 16, pp. p46-53, 2001.

[15] N. Gibbins, S. Harris and N. Shadbolt, "Agent-based semantic web services," in *Proceedings of the 12th international conference on World Wide Web*, Budapest, Hungary: ACM Press, 2003, pp. 710-717.

[16] D. Martin, A.J. Cheyer and D.B. Moran, "The Open Agent Architecture: A Framework for Building distribted Software Systems," *Appl.Artif.Intell.*, vol. 13, pp. 91-128, January-March 1999. 1999.

[17] K. Sycara, M. Paolucci, J. Soudry and N. Srinivasan, "Dynamic discovery and coordination of agent-based semantic Web services," *Internet Computing, IEEE*, vol. 8, pp. 66-73, 2004.

[18] L. Cabral, J. Domingue, E. Motta, T. Payne and F. Hakimpour, "Approaches to Semantic Web Services: An overview and comparisons," in *LECTURE NOTES IN COMPUTER SCIENCE*, vol. 3053, Open Univ, Knowledge Media Inst, Milton Keynes, Bucks, England Open Univ, Knowledge Media Inst, Milton Keynes, Bucks, England Univ Southampton, IAM, Southampton, Hants,England Ed. BERLIN: SPRINGER-VERLAG BERLIN, 2004, pp. 225-239.

[19] A. Ankolekar, M. Burstein, J.R. Hobbs, O. Lassila, D. Martin, D. McDermott, S.A. McIlraith, S. Narayanan, M. Paolucci, T. Payne and K. Sycara, "DAML-S: Semantic Markup ForWeb Services," in International Semantic Web Working Symposium (SWWS), 2001, pp. 348-363.

[20] J. Cardoso and A. Sheth, "Semantic e-workflow composition," *J Intell Inform Syst*, vol. 21, pp. 191-225, NOV. 2003.

[21] F. Curbera, R. Khalaf, N. Mukhi, S. Tai and S. Weerawarana, "The next step in Web services," *Commun ACM*, vol. 46, pp. 29-34, Oct. 2003.

[22] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi and S. Weerawarana, "Unraveling the Web services Web - An introduction to SOAP, WSDL, and UDDI," *IEEE Internet Comput.*, vol. 6, pp. 86-93, Mar-Apr. 2002.

[23] M. Paolucci, T. Kawamura, T.R. Payne and K. Sycara, "Semantic matching of Web services capabilities," in *Semantic Web - Iswc 2002*, vol. 2342; 2342, Berlin: SPRINGER-VERLAG BERLIN, 2002, pp. 333-347.

[24] R. Khalaf and F. Leymann, "On web services aggregation," in *Technologies for E-Services, Proceedings*, vol. 2819; 2819, Berlin: SPRINGER-VERLAG BERLIN, 2003, pp. 1-13.

[25] V. Tosic, B. Pagurek, K. Patel, B. Esfandiari and W. Ma, "Management applications of the Web Service Offerings Language (WSOL)," in *Advanced Information Systems Engineering, Proceedings*, vol. 2681; 2681, Berlin: SPRINGER-VERLAG BERLIN, 2003, pp. 468-484.

[26] V. Tosic, B. Esfandiari, B. Pagurek and K. Patel, "On requirements for ontologies in management of Web Services," in *Web Services, E-Business, and the Semantic Web*, vol. 2512; 2512, Berlin: SPRINGER-VERLAG BERLIN, 2002, pp. 237-247.

[27] J. Hendler, "Agents and the Semantic Web," *Intelligent Systems, IEEE [See also IEEE Intelligent Systems and their Applications]*, vol. 16, pp. 30-37, 2001.

[28] A. Hevner, S. March, J. Park and S. Ram, "Design Science in Information Systems Research," *MIS Quarterly*, vol. 28, MArch 2004. 2004.

[29] J. Nunamaker, M. Chen and T. Purdin, "System Development in Information Systems Research," *Journal of Managemennt Information Systems*, vol. 7, pp. 89-106, 1991. 1991.

[30] S. March and G. Smith, "Design and Natural Science Research on Information Technology," *Decis.Support Syst.*, vol. 15, pp. 251-266, 1995. 1995.

[31] H.A. Simon, The Sciences of the Artificial, MIT, 1996.

[32] W3C, "Web Ontology Language", "http://www.w3.org/2004/OWL/ ", 2005.

[33] J. Gray, "Distributed Computing Economics", MSR-TR-2003-24, "http://research.microsoft.com/research/pubs/view.aspx?tr_id=655", 2003.

[34] C.L. Forgy, "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problems," *Artif.Intell.*, vol. 19, pp. 17-37, 1982.

[35] S.A. Ludwig and P. Van Santen, "A Grid Service Discovery Matchmaker based on Ontology Description," in Euroweb 2002.

[36] S.A. Ludwig, "Flexible Semantic Matchmaking Engine," in 2nd IASTEP International Conference on Information and Knowledge Sharing (IKS), 2003.

[37] OWLJessKB, "http://edge.cd.drexel.edu/assemblies/software/owljesskb/", 2005.

[38] JESS, Java Expert System Shell, "http://herzberg.ca.sandia.gov/jess/", 2005.

[39] D. Fensel, F. Baader, M.C. Rousset and H. Wache, "Heterogeneous information resources need semantic access," *Data Knowl.Eng.*, vol. 36, pp. 211-213, Mar. 2001.