State Specialization in a Service Discovery Ontology: A Financial Services Business Grid

David Bell
Department of Information Systems and Computing
Brunel University
UK
David.Bell@brunel.ac.uk

Simone A. Ludwig
School of Computer Science
Cardiff University
UK
Simone.Ludwig@cs.cardiff.ac.uk

Mark Lycett
Department of Information Systems and Computing
Brunel University
UK
Mark.Lycett@brunel.ac.uk

Abstract

Investment Banking requires a diverse set of supporting systems in order to operate in a range of markets from corporate mergers to trading options on weather. The challenge to this community is the ability to adapt to new business requirements in an effective manner, utilizing their network of capabilities in a flexible and dynamic way. A semantic approach to capability discovery can combine many strategic perspectives in a pragmatic yet easily usable form. The use of richer explicit knowledge, that is system readable, provides the basis for discovering capabilities on this exemplar Business Grid - "the grid of services". Current research of semantic capability description in the Grid community has tended to focus on resource selection rather than service discovery. This research explores the practical usage of Grid services in the financial market sector. The approach demonstrates the need for distributed and phased semantic service discovery (with capabilities described and stored in a dynamic ontology).

1 INTRODUCTION

Investment banks house a diverse set of systems segregated by product, process or geographical focus. The product range include Foreign Exchange, Interest Rates, Fixed Income (Bonds), Commodities (from Oil to Weather) and Derivatives. Action is then required to bring together capabilities resident in several products, processes or geographical focused systems. The challenge to these organizations is to be able to adapt to new business requirements in an effective way, utilizing their capabilities in a flexible manner. The size of the inventory directs this research away from centralized knowledge engineering and use; and toward strategies for service knowledge segmentation. Focusing ones thinking on process and performance directs this analysis.

Grid infrastructure, with its Web Services layer, provides an infrastructure for exposing stateless and stateful organizational capabilities for re-use and novel re-configuration. The logical next step, with a service oriented inventory, is to identify practical methods for the discovery of these capabilities in the support of sporadic business need. Existing methods such as UDDI (Universal Description Discovery and Integration) [1] and MDS (Monitoring and Discovery Service) [2] were discounted due to the limitation of a string matching syntactic approach [3]. We have chosen a semantic approach to service discovery; investigating the practicalities of semantic technology in Financial Services (FS). The experiments explore the performance impact of top down conceptual models and bottom up state models; with results directing a novel phased combination of the two ontology formation mechanisms.

The remainder of this paper is organized as follows. In section 2 related work which motivates the research executed is presented. Section 3 shows the financial service architecture and describes the components used. In section 4 the implementation of the prototype is described with the tools used. Section 5 shows the performance graphs for two phases of experimentation. A conclusion is given in section 6 which summarizes the findings and suggests some further improvements.

2 RELATED WORK

Many general strategies have been proposed to address the issues of adapting to new business requirements in an effective way, utilizing these capabilities in a flexible manner. Social sciences have proposed organizational learning (OL), with tacit-explicit conversion of knowledge. This focus on knowledge flow, whilst often not computer readable, provides a basis for supporting improved inter silo decision making, clearly highlighted in Nonaka's knowledge spiral [4]. Alternatively, technology organizations have recently refocused on the use of workflow to orchestrate the execution of capabilities within the system inventory (an example being BPEL4WS [5]). One problem here is that no single

person, or pre-formed grouping, can bridge the wide conceptual knowledge gap between the macro and micro perspectives. Chen et al. [6] add, "such initiatives (WSFL, XLANG, BPEL4WS) generally focus on representing service compositions where the flow of process and the bindings between services are known a priori…". The upfront knowledge needed to create workflows for new, sporadic requirements is fragmented within the organization making the complete, top-down modeling of flows extremely difficult.

Similar research in the Grid area was addressed by Deelman et al. [7] with their workflow generator and Tangmurarunkit et al. [8] with their resource selector. The workflow generator addresses the problem of automatically generating job workflows for the Grid. Deelman et al. have developed two workflow generators. The first one maps an abstract workflow defined in terms of application-level components to the set of available Grid resources. The second generator takes a wider perspective and not only performs the abstract to concrete mapping but also enables the construction of the abstract workflow based on the available components.

The ontology-based resource selector exploits ontologies, background knowledge, and rules for solving resource matching in the Grid. The aim being to overcome the restrictions and constraints of resource descriptions in the Grid. In order to make the matchmaking more flexible and also to consider the structure of VOs the framework consists of ontology-based matchmakers, resource providers and resource consumers or requesters. Resource providers periodically advertise their resources and capabilities to one or more matchmakers using advertisement messages. The user can then activate the matchmaker by submitting a query asking for resources that satisfy the request specification.

Even though research on workflow and semantic resource selection has been introduced to the grid, it has been resource centric and not focused on service-orientation. In the financial market sector the challenge is being able to adapt to new business requirements in an effective way, utilizing these capabilities in a flexible manner. A semantic approach to capability discovery can combine OL and IT (Information Technology) in a pragmatic yet easily usable form, using general service profile ontology to support heterogeneity. It was with the aim of evaluating this technology that a business oriented discovery architecture was designed. The use of richer explicit knowledge, that is system readable, provides the basis for discovering capability on the Business Grid – "the grid of services" [9]. Our own generalized ontological service model was chosen – combining top-down human and bottom-up computer derived knowledge. OWL-S [10] was discounted due to (a) the impracticality in searching many distinct service profiles, (b) our need to reason across the whole inventory is better supported by a combined profile ontology and (c) discovery does not require much of the synthesis oriented baggage.

3 FINANCIAL SERVICE ARCHITECTURE

The business architecture of financial service applications are often segmented by product, process or geographic concerns; impacting service architecture design. Application re-use strategies will align behind the appropriate segment and additional development will be needed to close any recognizable gaps. Challenges may occur when factors dictate a particular strategy change or with misalignment to underlying segments. This may involve moving into a new market or developing a new business product from existing parts. The brittle nature of application integration is then replaced with that of brittle service integration. This is often caused by the static nature segmentation, service decomposition and registries. An alternative approach is to adopt a loosely coupled view of service description and categorization allowing more dynamic service groupings. It is with this strategy in mind that application system capabilities were replicated as stateful grid services in order to provide virtual services ready for exploitation via re-use or re-configuration.

In order to investigate the prototype, existing capabilities from a group of systems spanning product, process and geographical dimensions were extracted. The capabilities were explicitly described using OWL (W3C Web Ontology Language [11]) – see section 3. These same capabilities were exposed as grid services. In order to investigate dynamic discovery, three use cases were chosen to explore the applicability of semantic searching. (1) Searching for trades executed with a particular counterpart, (2) Valuing a portfolio of interest rate derivative products and (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. Extending our human derived ontological

model with state knowledge is investigated with three further use case – each involving various granularities of business state. The three state use cases comprise: (A) Risk Management where 10 books each hour are cached in the underlying grid service, (B) Pricing a Low Volume Product where 40 new trades per hour are cached in the underlying grid service and (C) Pricing a High Volume Product where 200 new trades per hour are cached in the underlying grid service.

3.1 Requirements

The use of a standard registry usually involves syntactically matching a request to an available service [3]. Furthermore, constraints imposed by specific information models on what and how service information can be stored can also limit applicability. The global financial organization has common terminology at the generalized level, which becomes more specialized when describing capability within a product, process or geographical context. These differences, together with relationships, need to be made explicit if capabilities are to be better identified.

3.2 Description of FS Service Components

Geographical coverage of the Investment Banking systems motivated the high level of middleware and knowledge flexibility. The matching algorithm may be deployed on one of four tiers; the client, a front line hosting server, a grid node or with the knowledge base.

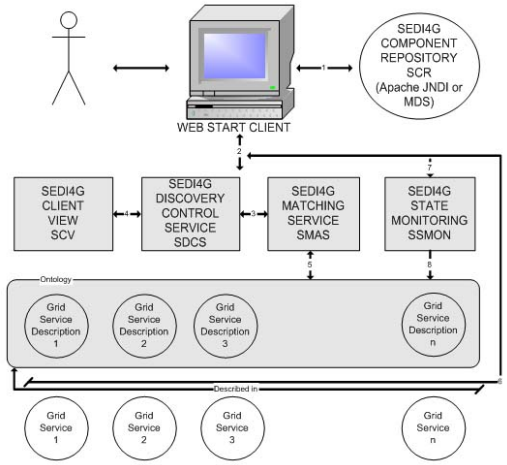

Figure 1. SEDI4G Architecture

The Semantic Discovery for Grid Services architecture (SEDI4G) (depicted in Figure 1), influenced by the changing network economics highlighted by Jim Gray [12], allows placement of the discovery services and knowledge across the network with associated selection and combination. Several FS applications where mined for capabilities (e.g. Object methods in a C++, .NET or Java based systems). Capabilities were exposed capabilities as stateful grid services. An example is trade pricing of Interest Rate Swap and Interest Rate Option products; resident in both trading and risk management systems. 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 resulted in certain terms requiring additional analysis. An example is the use of the term "trade" to describe the various states of the trade – planned, new, live and dead. These class specializations were recognized and made explicit manually in the first phase. The second phase of investigation extracted dynamic state without human intervention. The discovery process begins by identifying which components are required to carry out a semantic search. This choice involves placement concerns (represented by the grey flexible services and data in Figure 1). Thus, Step 1 involves the selection of which discovery control service (SDCS), knowledge base and matching service best fit the user requirement. This information is sent to SDCS together with the search parameters (2). SDCS then calls the KB based matching service (3) that in turn loads the KB and rules (5). The matching is carried out and returned to SDCS for use in one of the client components (4). The SDCS service can optionally provide the resource properties, the dynamic state of each service, alongside the service choices (6). These same properties are monitored by the SSMON service (7). This monitoring service contains simple heuristics that determine when a specialized description is required (8). The initial specialization rules are time based, an example being a stable state for x minutes.

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

Figure 2. 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 [13]. During the search request the ontology is reasoned depending on the specified Jess queries and the matched services are returned. Figure 2 shows in more detail how the matching algorithm works.

4 IMPLEMENTATION

The prototype system was implemented using Protégé (with OWL Plugin) and used to build three service KBs – OWL files (small, medium and large). The discovery algorithm is a grid enabled web service. The control web service is a Java web service that 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.

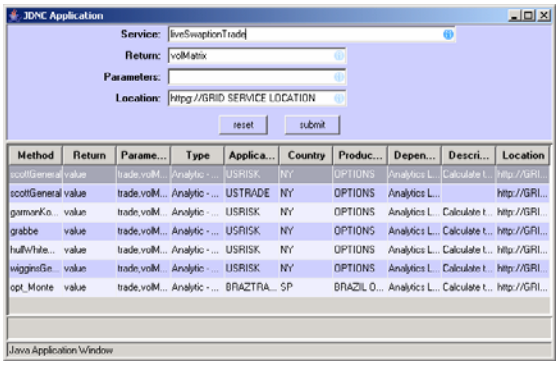

Figure 3. Service Search and Selection

The lack of common service discovery systems showed a lack of expressive service description and the lack of infrastructures for defining common service descriptions across enterprises and customers [14]. This is supported 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 [15]. The flexibility in capability categorization and multi-placement support address many of the limitations exhibited by static data models and centralized repositories.

OWLJessKB [16] was used for the service 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. It then inserts these triples as facts into the Jess knowledge base [17]. With some 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 within OWL files, allowing users to query it. It leverages the existing RDF API to read in the OWL file as a collection of RDF triples. How the service discovery process works is shown next. Figure 4 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 5 is invoked, returning results to the GUI (see Figure 3).

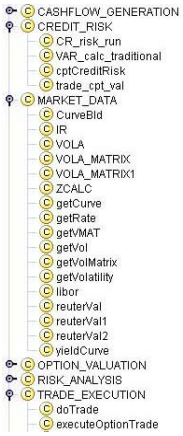

Figure 4. FS Ontology Structure

The returned classes are then taken and another Jess query for subsumption is invoked for each return class. This returns the services `getTradeByID`, `getTradeByDate`, `getTradeByType` etc. The sub-classing query finds the other three related services.

```
(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 5. Jess Query – Property of Class

5 PERFORMANCE MEASUREMENTS

Performance measurements were carried out using the distributed discovery architecture presented, utilizing various topologies and ontology sizes in order to determine size constraints imposed by using this richer service ontology. These measurements are

described in phase 1. In phase 2 performance measurements are undertaken to monitor and record service state for specialized service discovery.

5.1 Phase 1

The prototype system used three ontologies (summarized in Table 1) 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. The following setup was chosen. The three use cases mentioned in section 2 had the following query attributes:

1. *Finding a trade executed against a particular countparty [Query="tradeList",”cpt”]*
2. *Finding a capability to value a book of interest rate products [Query="rateVector", ”curveStructure”, ”doneTrade”]*
3. *Finding a capability to value an option trade [Query="liveSwaptionTrade", ”volMatirx”]*

The initial testing phase compared three differing ontology sizes (Table 1) when used to execute three use case searches.

	Small Ontology	Medium Ontology	Large Ontology
No. of classes	57	114	228
No. of properties	23	46	92

Table 1. Phase 1 Ontology Comparison

The results can be seen in Figure 6. The timings in 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). They are taken from services that have been initialized.

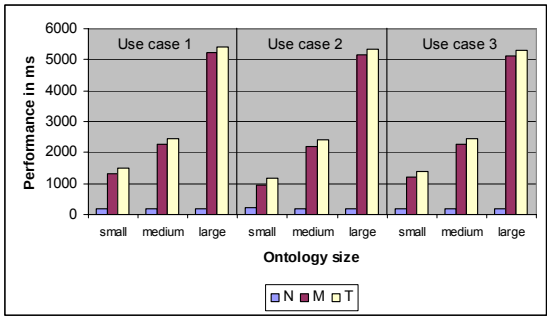

Figure 6. Ontology Size Results

The results highlight the near linear degradation in performance exhibited by the search. In a dynamic search context, ontologies larger than around 500 classes become impractical. Investment banks – with thousands of capabilities – require alternatives to centralization of service knowledge.

5.2 Phase 2

Recognizing the value of capability state in specialized sub-class definitions, warrants practical impact analysis. The second testing phase simulates the three stated dynamic ontology growth scenarios (which cause the number of classes to grow by (A) 80, (B) 320 and (C) 1600). The aim is to then understand the practicalities of state integrations and derive pre-discovery heuristics. The resulting ontologies were compared using the same three use case queries (see 5.1). The results are shown in Figures 7.

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

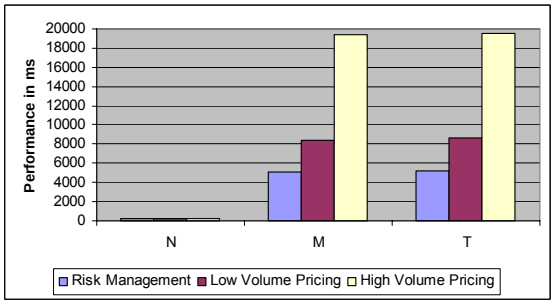

Figure 7. Combined Growth Results

An 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 architecture. 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. They are then selected in pre-discovery when a query attribute contains a state identifier. The linear growth in query performance does highlight that small service ontologies with minimal change in state are capable of remaining in a single KB. The use and performance limits of state are clearly stated.

The post-discovery heuristics expose additional relevance in the exploration of cost-benefit and associated payoff functions of service state: *(Cost-Benefit)*Importance*. Recognizing cost and benefit will differ depending on the context.

For example, in Sao Paulo the network usage will take precedence whereas in New York it is system performance. This requires an understanding of what is important and the associated costs and benefits of each specific state. Further research on dynamic ontology selection is required.

6 CONCLUSION

This resulting SEDI4G service discovery system and performance results show that dynamic service discovery is easily and flexibly supported with JESS based semantic search. Ontology sizes of up to 200-300 classes (the large ontology test case) yield adequate performance. The performance degrades linearly with the number of classes. Likely ontology sizes within an Investment Bank extend into thousands of classes and direct the second phase investigation of specialized KBs and other optimization methods.

The novel use of bottom-up resource state heuristics to dynamically extend the top-down service ontology, whilst degrading search performance, must be considered in relation to the value of selecting such specialized resources. The combination of the domain size and the inclusion of valuable service state emphasize the benefits of a discovery process that is both architecturally distributed and phased. This need to recognize the phasing and distribution of service knowledge is the key contribution of this paper, as the implications to the global business capability knowledge base is one of poor performance or over generalization if not appreciated. The paper demonstrated the suitability of pre- and post-discovery phases (and associated rule sets) in that they provide knowledge selection and specialization functions.

7 REFERENCES

[1] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith and S. Tuecke, “A Directory Service for Configuring High-Performance Distributed Computations”. Proceedings of the 6th IEEE International Symposium on High-Performance Distributed Computing (HPDC-6), 1997.

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

[3] L. Li and I. Horrocks, “A Software Framework for Matchmaking Based on Semantic Web Technology”, International Journal of Electronic Commerce, 8 (4). 39-60.

[4] I. Nonaka, “The Knowledge-Creating Company”, Harvard Business Review, 69 (6). 96, 1995.

[5] BPEL4WS - Business Process Execution Language for Web Services Version. http://www-128.ibm.com/developerworks/library/ws-bpel/.

[6] L.M. Chen, N.R Shadbolt, C. Goble, F. Tao, S.J. Cox, C. Puleston and P.R. Smart, “Towards a Knowledge-based Approach to Semantic Service Composition”, Proceedings of 2nd International Semantic Web Conference (ISWC2003), 2003.

[7] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, A. Lazzarini, A. Arbree, R. Cavanaugh, S. Koranda, Mapping Abstract Complex Workflows onto Grid Environments, Journal of Grid Computing, Vol. 1, No. 1, pp 9--23, 2003.

[8] H. Tangmunarunkit, S. Decker, C. Kesselman, "Ontology-based Resource Matching in the Grid - The Grid meets the Semantic Web", In the proceedings of the First Workshop on Semantics in Peer-to-Peer and Grid Computing (SemPG03). In conjunction with the Twelfth International World Wide Web Conference 2003. Budapest, Hungary. May 2003.

[9] F. Leymann and K. Guntzel, “The business grid: Providing transactional business processes via grid services”, Proceedings of Service-Oriented Computing (ICSOC2003), 2003.

[10] OWL-S, OWL Services. http://www.daml.org/services/owl-s/.

[11] W3C Web Ontology Language. http://www.w3.org/TR/owl-ref/.

[12] J. Gray, “Distributed Computing Economics”, Microsoft Research, MSR-TR-2003-24, 2003.

[13] C.L. Forgy, “Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem”. Artificial Intelligence, 19(1982) 17-37.

[14] S.A. Ludwig et al., “A Grid Service Discovery Matchmaker based on Ontology Description”, Proceedings of 2nd International EuroWeb2002 Conference, Oxford, UK, 2002.

[15] S.A. Ludwig, “Flexible Semantic Matchmaking Engine”, Proceedings of 2nd IASTED International Conference on Information and Knowledge Sharing (IKS), AZ, USA, 2003.

[16] OWLJessKB. http://edge.cs.drexel.edu/assemblies/software/owljesskb/.

[17] JESS, Java Expert Systems Shell. http://herzberg.ca.sandia.gov/jess/docs/61/index.html.