Grid Service Discovery in the Financial Markets Sector

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

Abstract. *Investment Banking requires a diverse system set in supporting a range of markets from bonds 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, dynamic way. A semantic approach to discovery can be used in a pragmatic, practical manner. 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". This design research project focuses on the utilisation of disparate knowledge during discovery.*

Keywords. Grid Services, Ontology, Discovery

1. Introduction

Investment Banking software is a diverse domain to investigate – with many market based categories ranging from Mergers to the Global Markets. The product range include Foreign Exchange, Interest Rates, Fixed Income (Bonds), Commodities (from Oil to Weather) and Derivatives. The Derivative product uses one or more of the base products as an underlying; a complex example being a total return swap which combines the characteristics of a bond with an interest stream of a swap. The trade then embarks on a lifecycle involving state transitions caused by various process enactments. Processes range from pre-trade validation between organizations to trade maturation. The resulting inventory contains many systems that focus on particular products, processes and geographical segments. Technological challenges arise when new products combine existing ones or with synergistic process deployment. The challenge to these organizations is to be able to adapt to new business requirements in an effective way, utilizing these capabilities in a flexible manner. This research aims to address these challenges by loosening the coupling to the systems capability inventory allowing a more configurable approach to utilization.

Grid infrastructure, with its Web Services layer, provides an infrastructure for exposing stateless and stateful organizational capabilities, a difficulty in large global organisations. The logical next step is to identify practical methods for the discovery of these capabilities when supporting this ad-hoc business capability identification. 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]. A semantic approach was chosen because of the need to understand the complex relationships between system capabilities. Especially when exposed as stateful grid services – with complex state and sub-state. The design focus on ontology artifacts in relation to the whole discovery process is the central theme – exploring search performance with practical Financial Services (FS) knowledge topologies.

The remainder of this paper is organized as follows. Section 2 covers related work. Section 3 shows the financial service architecture. In section 4 the implementation of the prototype is described. Section 5 contains the performance analysis. A conclusion is given in section 6 which summarizes the findings.

2. Related Work

Many strategies have been proposed to address the issues of adapting to new distributed business requirements either through the design of knowledge flows as seen in Nonaka's knowledge spiral [4] or the technology focus on explicit workflow design (BPEL4WS [5]). Both approaches do not directly address an environment of changing business requirement and strategy across business silos. The explicit knowledge created from experience is valuable, but not in a system usable format and as such cannot be readily utilized when searching for existing capability electronically. Alternatively, the workflow approaches provide an explicit definition of how a capability is used at both a macro (the workflow) and micro level (the service). 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…".

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. They 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 system operates in the application domain and chooses application components based on the application metadata attributes.

The ontology-based resource selector exploits ontologies, background knowledge, and rules for solving resource matching in the Grid 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. The query is then processed by the TRIPLE/XSB deductive database system [9] using matchmaking rules, in combination with background knowledge and ontologies to find the best match for the request.

The use of richer explicit knowledge, that is system readable, provides the basis for discovering capability on the Business Grid – "the grid of services" [10]. This diversity of human and system focus is bridged within this research by focusing on the use of multiple, co-existing service ontologies as opposed to more centralized or homogenized resource centric approaches.

3. Financial Service Architecture

Business and technical architecture of financial service applications may be segmented by product, process or geographic concerns. Segmented inventories make inter silo re-use difficult. The brittle nature of application integration is then replaced with that of service integration. An alternative approach is to adopt a loosely coupled inventory – containing differing explicit capability knowledge.

In order to investigate the practical use of capability knowledge; capabilities from a group of systems that spanned the product, process and geographical dimensions were extracted. The capabilities were explicitly described using OWL (W3C Web Ontology Language [11]) – see section 3 – and exposed as grid services. Three use cases were chosen to explore the use of semantic searching:

- *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 use-cases were chosen because they provide examples of three distinct patterns of use – aggregation, standard selection and multiple selection.

The Semantic Discovery for Grid Services architecture (SEDI4G) (depicted in Fig. 1) allows placement of the discovery services across the network. Several applications from this domain where mined for capabilities (e.g. methods and parameters). These capabilities were then stubbed as grid and web services. With examples such as methods for the trade pricing of Interest Rate Swap and Interest Rate Option products; resident in both trading and risk management systems. The use of *de facto* business terminology and object generalization can result 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 specializations were recognized and made explicit when describing the services in ontological form as this gave greater selection accuracy.

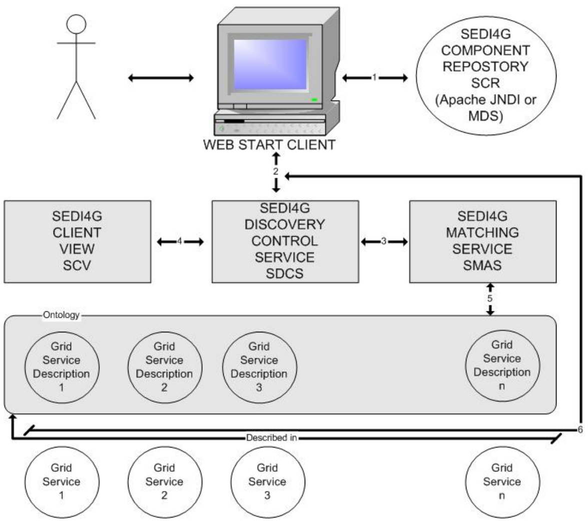

SEDI4G
COMPONENT
REPOSTORY
SCR
(Apache JNDI or
MDS)
1
WEB START CLIENT
2
SEDI4G CLIENT VIEW SCV
4
SEDI4G DISCOVERY CONTROL SERVICE SDCS
3
SEDI4G MATCHING SERVICE SMAS
5
Ontology
Grid Service Description 1
Grid Service Description 2
Grid Service Description 3
Grid Service Description n
Described in
6
Grid Service 1
Grid Service 2
Grid Service 3
Grid Service n

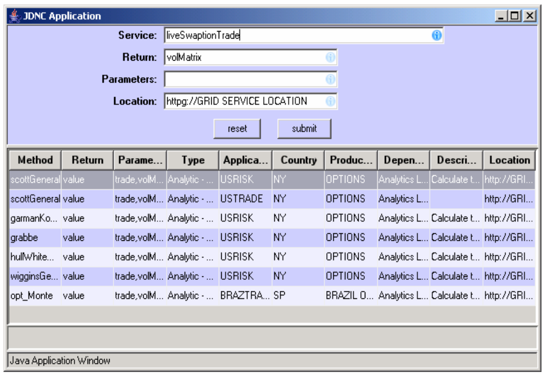
Figure 1. SEDI4G Architecture

The discovery process begins by identifying components required to carry out a semantic search. This choice involves mainly placement concerns (represented by the grey flexible services and data in Fig. 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).

As the matching algorithm is based on a semantic method using an ontology and a reasoning engine, the background to the subject is introduced. The subject of ontology, a term borrowed from philosophy [12], is the study of the categories of things that exist or may exist in some domain [13]. An ontology 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 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 [14]. During the search request the ontology is reasoned depending on the specified Jess queries and the matched services are returned.

4. Implementation



Service:	liveSwaptionTrade
Return:	volMatrix
Parameters:	
Location:	httpg://GRID SERVICE LOCATION

Method	Return	Parame...	Type	Applica...	Country	Produc...	Depen...	Descri...	Location
scottGeneral	value	trade,volM...	Analytic - ...	USRISK	NY	OPTIONS	Analytics L...	Calculate t...	http://GRI...
scottGeneral	value	trade,volM...	Analytic - ...	USTRADE	NY	OPTIONS	Analytics L...		http://GRI...
garmanKo...	value	trade,volM...	Analytic - ...	USRISK	NY	OPTIONS	Analytics L...	Calculate t...	http://GRI...
grabbe	value	trade,volM...	Analytic - ...	USRISK	NY	OPTIONS	Analytics L...	Calculate t...	http://GRI...
hullWhite...	value	trade,volM...	Analytic - ...	USRISK	NY	OPTIONS	Analytics L...	Calculate t...	http://GRI...
wigginsGe...	value	trade,volM...	Analytic - ...	USRISK	NY	OPTIONS	Analytics L...	Calculate t...	http://GRI...
opt_Monte	value	trade,volM...	Analytic - ...	BRAZTRA...	SP	BRAZIL O...	Analytics L...	Calculate t...	http://GRI...

Figure 2. Service Search and Selection

A prototype system was implemented using Protégé (with OWL Plugin) to build three service KBs – OWL files (small, medium and large). The discovery components are grid enabled web services. 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 Java Desktop JDNC data file.

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].

OWLJessKB [16] was used to read and query the OWL files. It inserts the RDF 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 of OWL files, and it allows users to query on that information. 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.

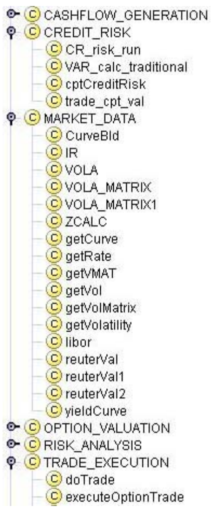


Figure 3. FS Ontology Structure

Fig. 3 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 Fig. 4 is invoked. The returned classes are then taken and the Jess query shown in Fig. 5 is invoked for each return class. This returns the services `getTradeByID`, `getTradeByDate`, `getTradeByType` etc. The sub-classing query finds the other three related services, which is basically done with the subsumption relation which defines a hierarchy of concepts.

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

Returned services are then displayed in the GUI (Fig. 2).

```
(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 5. Jess Query – Sub-Classing

5. Performance Measurements

The prototype system used three ontologies (summarized in Table 1) to investigate the impact of KB size on semantic search performance. The ontology growth will be driven 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:

- *Use case 1: Finding a trade executed against a particular countparty [Query="tradeList", ”cpt”]*

- *Use case 2: Finding a capability to value a book of interest rate products [Query= "rateVector",”curveStructure”,”doneTrade”]*
- *Use case 3: Finding a capability to value an option trade [Query="liveSwaptionTrade", ”volMatrix”]*

Table 1. Ontology Comparison

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

Two sets of measurements were taken – ontology size (Fig. 6) and placement (Fig. 7).

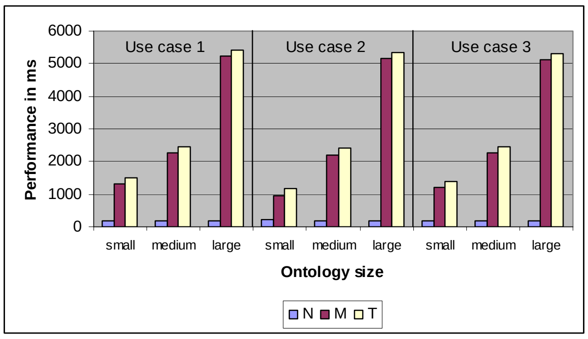


Figure 6. Performance Test - Local

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. As seen from the measurements, the longest timings occur during the matchmaking process, due mainly to the Rete algorithm.

Table 2. Average Result of Local Performance

Ontology Size	Perf. of N in ms	Perf. of M in ms	Perf. of T in ms
Small	196	1158	1354
Medium	188	2255	2443
Large	181	5171	5353

Table 2 summarizes the average results of the first set of measurements. The measurements taken show an average variation of 8ms, 67ms and 63ms of the time taken for N, M and T respectively. This is as expected as the time for N is relatively small in comparison to M and T.

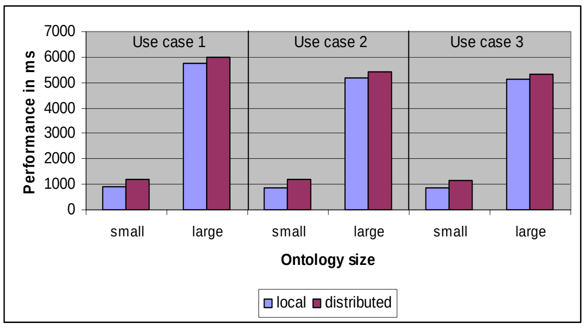


Figure 7. Performance Test – Local vs. Distributed

Fig. 7 shows the performance measurements of three use cases using local and distributed KBs. Comparing the results quantitatively reveals that the communication overhead has a bigger impact on the small ontology than on the large ontology. Measured for this overhead were 24%, 28% and 24% for the small ontology and 5%, 4% and 3% for the large ontology. This is due to the fact that the time taken loading the small ontology and the application of the rules were much smaller than for the larger ontology.

Table 3. Average Local vs. Distributed

Ontology Size	Local Performance	Distributed Performance
Small	883ms	1182ms
Large	5362ms	5576ms

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. The overhead increases to 25% for the small distributed KB.

6. Conclusion

This resulting system and performance results show that dynamic re-use of service capabilities is easily supported by the 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 grows linearly with the number of classes. Likely ontology sizes within an Investment Bank extend to many thousands of classes directing further work on specialized KBs and other optimization methods. Distributed KB support is a key characteristic of the SEDI4G architecture, allowing for many KBs to exist within its search scope (a prerequisite when thousands of classes exist across the organization).

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", Proceedings of the International Semantic Web Conference, Budapest, Hungary. May 2003.

[9] The XSB Research Group. http://xsb.sourceforge.net.

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

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

[12] T.R. Gruber, "A Translation Approach to Portable Ontology Specifications", Knowledge Acquisition, Vol. 5, No. 2, 1993.

[13] J.F. Sowa, "Ontology, Metadata, and Semiotics, Conceptual Structures: Logical, Linguistic, and Computational Issues", Lecture Notes in AI #1867, Springer-Verlag, Berlin, pp. 55-81, 2000.

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

[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.