Computing and Informatics, Vol. 28, 2009, 635–654

DESIGN AND DEVELOPMENT OF FINANCIAL APPLICATIONS USING ONTOLOGY-BASED MULTI-AGENT SYSTEMS

Weir Ying, Anjalee Sujanani

School of Computer Science and Engineering
School of Information Systems, Technology nd Management
University of New South Wales, Australia
e-mail: weir.ying@gmail.com, a_sujanani@yahoo.com

Pradeep Ray

School of Information Systems, Technology and Management
University of New South Wales, Australia
e-mail: p.ray@unsw.edu.au

N. Paramesh

School of Computer Science and Engineering
University of New South Wales, Australia
e-mail: paramesh@cse.unsw.edu.au

Damien Lee, Ramaprasad Bhar

School of Finance, University of New South Wales, Australia
e-mail: lee.damo@gmail.com

Manuscript received 12 October 2006; revised 29 April 2009

Communicated by Jacek Kitowski

Abstract. Researchers in the field of finance now use increasingly sophisticated mathematical models that require intelligent software on high performance computing systems. Agent models to date that are designed for financial markets have their knowledge specified through low level programming that require technical expertise in software, not normally available with finance professionals. Hence there is a need for system development methodologies where domain experts and researchers and can specify the behaviour of the agent applications without any knowledge of the underlying agent software. This paper proposes an approach to achieve the above objectives through the use of ontologies that drive the behaviours of agents. This approach contributes towards the building of semantically aware intelligent services, where ontologies are used rather than low level programming to dictate the characteristics of the agent applications. This approach is expected to allow more extensive usage of multi-agent systems in financial business applications.

Keywords: Ontology, multi-agent systems, financial services domain

1 INTRODUCTION

Financial market is a mechanism that allows people to trade financial securities (such as stocks and bonds) and other commodities. Financial services allow these markets to operate by providing ways for communication and trade. Some of the modern financial applications utilise intelligent mechanisms for transactions over the Internet that connects geographically distributed entities. A multi-agent architecture supports intelligent mechanisms through software agents that have the capability to conduct conversation-like contract negotiations, which are critical during coordination of business transactions. Agents also have the capability to build up its intelligence and knowledge-base over time, allowing intelligent services. Finally, a software agent by definition has the ability to act autonomously and adapt based on changes in the situation.

In order to gain a collective understanding of financial markets, it is important to observe and investigate the relationships between trends and characteristics across different markets. However, due to the complex conglomeration and distributed nature of financial domain information, the majority of such analysis is carried out at a low level, requiring extensive knowledge of programming languages. This can be problematic for those financial information users, researchers and analysts who do not have the expertise required for carrying out complex development in these languages.

In this paper, we describe the processes we undertook in the design and development of Financial Market Builder (*FinBuilder*) – an application that facilitates user customisation of agent interaction within the financial domain. We believed that the introduction of an ontology-driven infrastructure for use in the financial domain would enable experimentation activities such as pre-trade and behavioural

analysis of financial markets to be conducted by end-users at a more intuitive level than is currently possible.

FinBuilder is currently built on an agent platform, as our main aim was to investigate the combining of ontology semantics with a distributed application. The application is deployable on the web, thus the concepts covered are applicable to a web service platform. The integration of semantics with web services can be used to assimilate data from different domains or sources, and to infer information based on background knowledge of a domain.

In the following sections, we introduce financial multi-agent systems and their current downfalls. Subsequent sections will illustrate the benefits of introducing ontology into these systems. The FinBulder development process and architecture will be discussed. This is followed by the results of testing and evaluation of the application, and a summary of the research contribution concludes this paper.

2 MULTI-AGENT SYSTEMS IN FINANCE

Traditional mathematical methods used to study financial market behaviour such as statistical analysis have been identified as having shortcomings such as the following [18]:

Description of macroscopic properties of a system, not the origin of these properties. This analysis involves studying financial data from different financial markets and then identifying regular patterns of the data statistics. It usually does not include examining the cause of such statistics.

They fail in situations where the assumptions of mathematical models are not valid. The majority of statistical methods and techniques involve assumptions such as normal population. However, in many cases these assumptions do not hold.

They do not adequately handle the heterogeneity of trade practices. Traditionally, the behaviours of traders have been described with mathematical models under equilibrium conditions that is not always the case. Traders, for example, display heterogeneity in their trade decision-making, interpretation of company announcements and market trends, and adaptive behaviours.

In dealing with the dynamics of collections of entities, agent-based models (multi-agent systems discussed in this paper) are better equipped to handle different kinds of global dynamics that can result from these entities significantly impacting each other through their interaction within changing environments.

However, of the financial agent systems described in the literature, we found that most of the agents in these models were intrinsically algorithmically linked, with mathematical functions dictating and modifying the agents' behaviours [4, 5]. That is, financial domain knowledge and business logic is implicit in the algorithms and embedded in the agent code. There do exist multi-agent systems where the knowledge is represented more explicitly at organizational levels and using ontologies in

other domains [6]. Furthermore, in the majority of the work surveyed, the agent infrastructures were closely coupled with the application domain knowledge required to dictate the agents' behaviours. By placing most of the explicit domain information within the agents themselves, any potential to re-use the multi-agent infrastructure in conjunction with different domains was destroyed. Hence we propose ontology-based multi-agent systems where the financial system behaviour can be controlled through financial ontologies (defned by financial analysts with little software expertise) that drive the underlying multi-agent software (designed by software experts).

3 ONTOLOGY-BASED MULTI-AGENT SYSTEMS

An ontology is a specification of the objects, concepts and entities that exist in a domain of interest and the relationships that hold among them. They have been used in the fields of artificial intelligence, information retrieval, natural language processing and knowledge engineering. In the domain of B2B markets, ontologies have been utilised to address interoperability problems that enterprise and e-commerce systems face when attempting to share information, due to differing configurations and communication standards [1]. Additionally, ontology mapping has been used to improve semantic translation between network management models, where multiple information languages define the same set of resources to be managed [2].

Domain specific ontologies define concepts in terms of semantics that are applicable to a certain area. They can contain rules defined in machine processable languages to perform automated reasoning. By defining domain ontologies as a common framework for specific requirements, ontology developers are able to reuse such frameworks and provide for application and information integration.

It was our aim to apply the ontological concept to this end in a domain where such work has hitherto been fairly sparse – the financial market domain. The financial landscape is complex and volatile by nature, making timely information about market trends critical to strategic success. As a result, the study of financial market behaviour exists as a consequential field of endeavour for researchers and financial analysts alike.

It was felt that ontologies would offer a solution for the management of information dissemination as the sharing of common domain concepts and relationships could bridge the different viewpoints of agents. In addition, the creation of an ontology relies to a greater extent on the knowledge of the domain of application – such as finance – than on programming knowledge. A final advantage of using ontology in capturing concept of trading is that reasoning engines can be used in the evaluation of trading strategies.

A survey of current work discovered few financial domain ontologies, and none of those found had been written with the purpose of utilisation in multi-agent systems in the manner proposed by our project.

For instance, in the stock market ontology of [7], the low-level design details that describe the elements, relationships and rules of a stock market domain are

presented. Though the paper focuses on the reusability of the ontology, it does not provide an application demonstrating how this could be made possible. In another two studies [8, 9], the authors propose the use of financial domain ontologies within a multi-agent system. However, the ontologies developed were used only as a common semantic interface for agents where domain knowledge still resides within the agents. Our aim is to allow end-users (i.e. financial domain experts) to modify the system without needing the knowledge required to program agent behaviour in low-level languages. Other studies of financial market ontologies mainly focussed on ontology mapping (such as mapping across different news sources or information formats) [10], however this is not the focus of our paper. We also saw projects such as [11] where ontology was introduced into multi-agent systems. Although these projects have similarities in their overall intended goals, they were explored in a different domain.

Thus, though the projects surveyed provided insights in developing ontologies within the financial services domain, they were lacking in some key benefits of using ontologies. We hence develop our own financial ontology.

3.1 Ontology Development

Ontology development methodologies are a series of steps defining a process in which an ontology can be created systematically. Because of the myriad of factors involved in developing an ontology, such as the purpose, intention and domain, finding a common methodology for ontology engineering is difficult [12]. The methodologies surveyed in [13] either did not provide details of building steps or was not domain specific. Based on analysis of currently available methodologies and commonly adopted steps we propose a five step methodology for the creation of our financial market ontology. Below is a summary of the high level steps involved.

Knowledge Acquisition. In this stage, we determined the scope, domain and purpose of the ontology to be created. Knowledge acquisition was a major challenge that was faced during the ontology design phase. This problem was simplified by our decision to scope down the ontology domain to focus primarily on stock markets. In order to perform knowledge acquisition, we firstly studied the market processes with focus on electronic trading. Further understanding was gained through collaboration with financial experts and acquiring draft sets of financial domain concepts and their relationships. We also examined financial market data including historical trade data, order details and market statistics in relation to company announcements.

Conceptualisation. This is the process needed to turn raw knowledge into clearly established concepts used to create our ontology. With constant interaction and consultation with financial experts, we structurally organized and conceptualized the raw information into the following aspects:

Concepts – These are keyword or phrase descriptions of the domain of our ontology – for example securities, portfolios, stock symbols, buyers, sellers and stock market prices.

Facts – Instances of some concepts may be facts. For example, it is a fact that the instance TLS, of a stock symbol, stands for TELSTRA CORPORATION (Australia's leading telecommunications company). These instances are usually populated after the ontology is complete.

Some examples of concepts that we have included in our minimal set of financial domain ontology include Portfolio, Stock, Company and MacroEvent, representing macroeconomic events. The macroeconomic events are further broken down into LossEvent, TakeOverEvent and ProfitDropEvent. They correspond to the following, respectively:

- announcements of company losses
- announcements of company takeovers
- announcements of drops in profit compared with the previous period.

Semantic Modelling. In this stage, we systematically modelled and enriched the meanings of concepts. Since the ontology is meant to be for agent consumption rather than for humans, the semantics of the concepts were modelled keeping the agent architecture in mind. For the sake of simplicity, a BDI agent [14] was assumed. The semantics of a concept was specified in terms of the behaviour required of the agent responsible for the task of trading. A semantic model for a particular concept essentially includes a set of attributes that characterize the meanings of the concept. These include:

Relationships – These are dependencies or connections between concepts – for example stocks are bought and sold by traders or stock portfolios belong to traders.

Constraints – The cardinality constraints on attributes of a concept.

Using the concepts mentioned previously, a partial branch of our financial domain ontology tree is illustrated in the following graphical representation.

Figure 1 presents a portion of our financial domain ontology where the concepts Portfolio, Stock, Company and MacroEvent are children of an abstract root concept called Concept. Additionally, the concepts TakeOverEvent, LossEvent and ProfitDropEvent are child concepts of MacroEvent. The dotted lines connecting concepts represent a non parent-child relationship. The relationships between the concepts Portfolio and Stock are 'containsStock' and the inverse 'isPartOfPortfolio'. Relationships for the concepts Company and Stock are 'isIssuedBy' and 'ownsStock'. For Company and MacroEvent the relationships are 'hasEvent' and 'eventBelongsTo'.

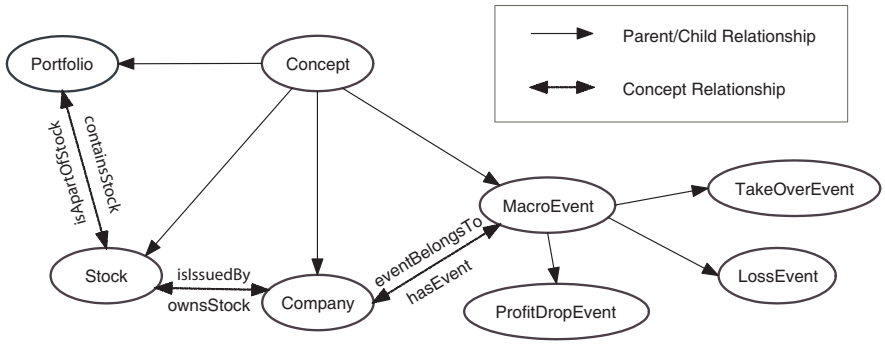


Fig. 1. Graphical representation of partial financial domain ontology

Knowledge Representation. Here, we formally encoded the semantics identified and captured in the previous steps. We used Protégé which stored the ontology internally as RDF for a number of reasons. Firstly, we felt the Protégé interface was both intuitive and user-friendly, not requiring a large amount of time to become familiar with. Secondly it contained a large number of plug-ins that enabled the user to extend the editor's core functionality. Some of the plug-ins that looked especially useful were the OntoViz Tab, which enabled the visualisation of Protégé ontologies and the XML Tab, which enabled Protégé ontologies to be extracted from XML files and XML files to be translated into Protégé ontologies. This could facilitate the depiction of the ontology in a more presentable manner.

The final deciding factor was that Protégé projects could be translated automatically into FIPA/JADE compliant ontologies using a tool called the Java Ontology Bean Generator [15]. In past work, ontologies have had to be manually translated into more restricted machine readable formats such as XML, database schema, or object oriented schema in order to bridge the communication gap between software agents and the actual ontology. As we had decided to use the JADE multi-agent environment [16] for implementation of *FinBuilder*, we felt that this added automation would be a great advantage, as it would remove the need to manually translate the ontology from the editor specification into an agent-understandable format. Using the editor, we encoded the financial ontology within the Protégé environment. Methods such as proposed by [17] were also considered. However, these methods were not as flexible or efficient when used for our implementation.

Validation. Normally this step precedes the knowledge representation step. We see a major benefit in postponing this step and exploit the tools associated with the ontology representation language (e.g. OWL/DL) to perform automatic checking of consistency. Any defect will lead back to the conceptualization step resulting in a cyclic ontology development process. Figure 2 illustrates a section of the

development as visualised within the Protégé environment. The left side of Figure 2 shows a snapshot of the ontology structure in a tree form while on the right, details of each concept, its attributes, relationships and constraints are displayed.

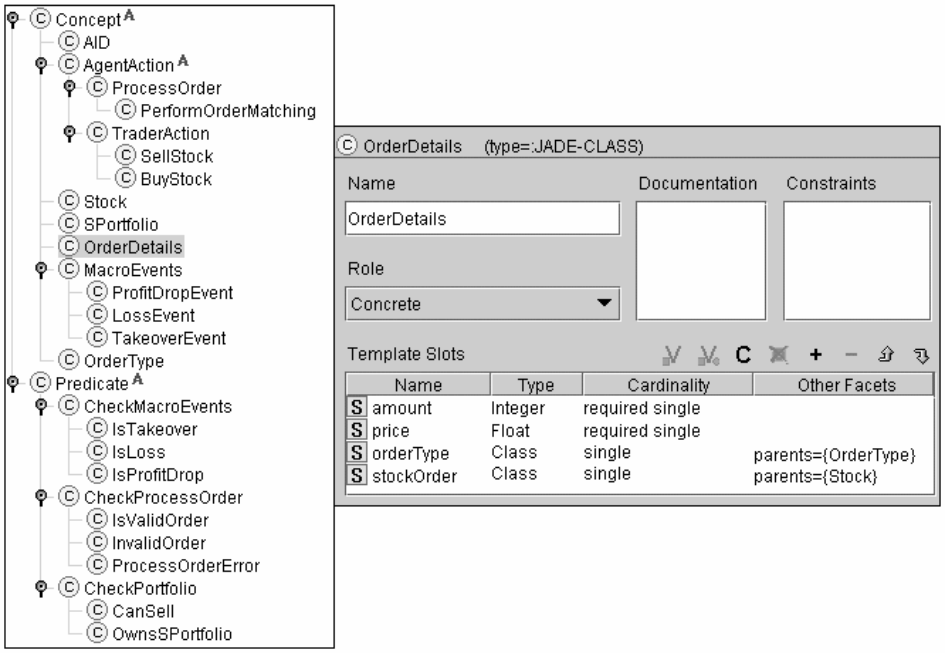


Fig. 2. Protégé ontology development snapshot

4 FINBUILDER

4.1 *FinBuilder* Architecture

Following the acquisition of financial market knowledge, the conceptual outline of the financial ontology was developed. The process of knowledge of semantic modelling and knowledge representation can then be done in Protégé. Verification and validation of the ontology can then follow. Figure 3 illustrates the overall ontology development and *FinBuilder* architecture.

We investigated a number of multi-agent platforms and decided to use the JADE framework as it had the greatest ontology support. The content reference model in JADE enabled ontologies subscribing to its model to be accessed by its agents. The model required the inclusion of low-level ontological elements – predicates, terms, concepts and agent actions. The Bean Generator Tool allowed us to generate a FIPA/JADE compliant ontology from the ontology specified in Protégé.

The complexity of the financial domain such as reactions to different financial events and trading decisions influenced the design of the behaviours of the *FinBuilder* agents. The input source was market data that was manipulated and fed into the *FinBuilder* tool. Interaction with the ontology directly enabled agents to understand and process the data feed and make trade decisions. This is illustrated in Figure 3 (adapted from [18]) by the external entity Trading Data Sources which produce the Market Data feeds.

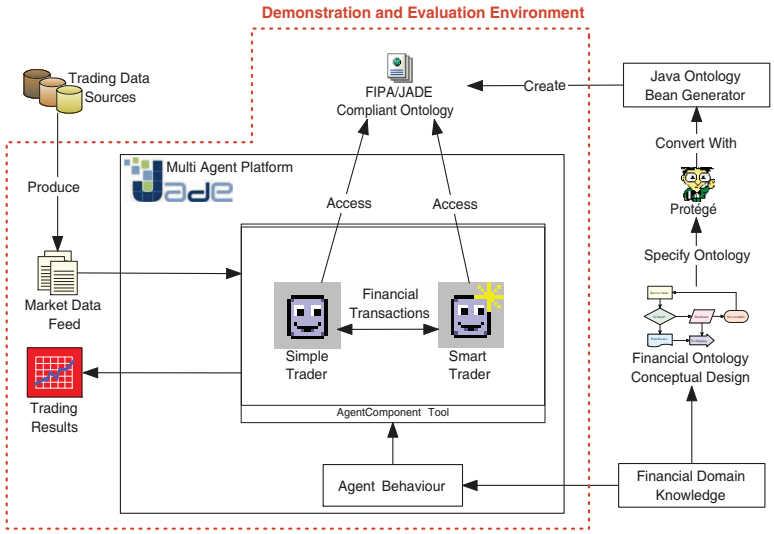


Fig. 3. Overall Architecture

4.2 Agent and Reasoning

We developed some scenarios with which to test the *FinBuilder* tool in this market environment. Figure 4 illustrates the agents that are created and the ontological layers each used for communication.

Order Agent: This agent is comparable to an electronic trading website that allows traders to submit buy and sell orders. The role of this agent is to facilitate an interface between the market and the traders.

In Figure 4, all trader agents are able to internally submit orders to this agent without the requirement of additional message translation as the agents are committed to the same ontology. An interface was created for entering external orders into the system. This allowed us to mimic market behaviour as we enter series of orders into the system and observe the agent's reaction to the change of the market state. Both order and order processing agents are apart of the market mechanism block as shown in Figure 4.

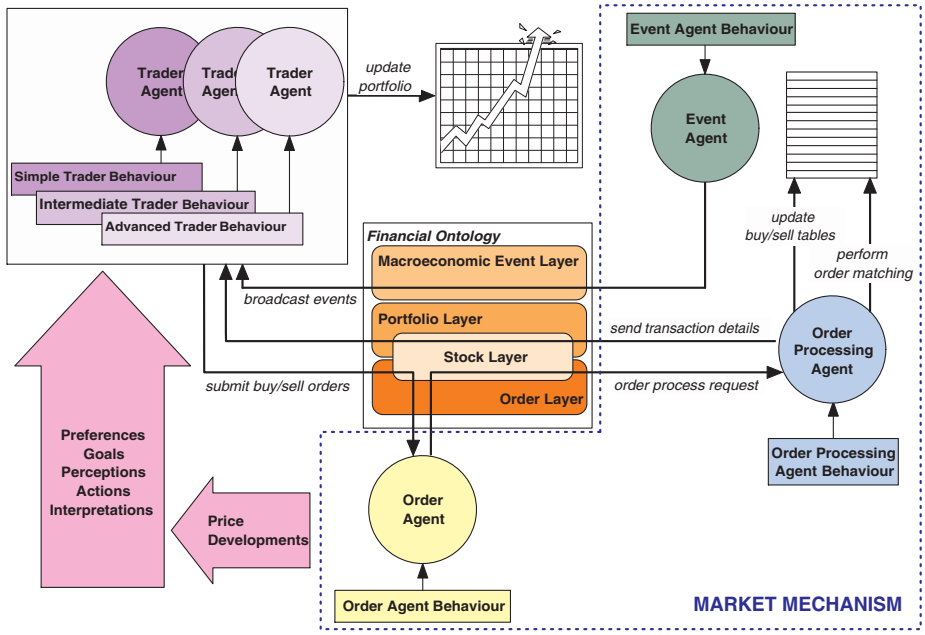


Fig. 4. Agent-Ontology Interaction [18]

Order Processing Agent: This agent represents the functions of a stock market trading engine. This agent performs tasks such as receiving new orders from the Order Agent and replies with confirmation of order submission. Order Processing Agent also updates the market buy and sell order tables by performing order matching. These tables display a continual listing of the current buy and sell orders – including the prices set for limit orders, the stock name and symbol, and the order quantity. Once a successful transaction is completed, the agent either removes the orders from the tables, or updates the buy and sell quantities displayed.

Event Agent: The Event Agent mimics the movement of stock prices in our artificial market by introducing macroeconomic events. The Event Agent represents these events, and disseminates announcements relating to companies to all traders. Macroeconomic events change both agent behaviour and stock prices. Communication between this agent and the trader agents is carried out through messages conforming to the macroeconomic event ontology layer. Each trader agent's reaction to these events varies according to its level of sophistication. The macroeconomic events considered are represented by the concepts in Figure 1.

Trader Agent: These agents represent traders in our stock market. Trader agents comprise the main entities of interest in the prototype. Through their perfor-

mance, the ability to simulate trading with the financial ontology can be evaluated. *FinBuilder* models the heterogeneity of stock market traders through three different trader agent types. These are:

Simple Agent – exhibits primitive trading behaviour.
Intermediate Agent – has moderately informed trading behaviour.
Advanced Agent – possesses sophisticated trading behaviour.

In order for meaningful comparison of agent performance to take place, each agent is initialised with an ownership of the same number and valuation of stocks. Each agent is also provided with a list of stocks that are interested in buying. This reflects real-world trading decisions to invest in technology stocks or blue chip stocks. For the purposes of better performance comparison, we decided to standardise the number of shares each agent buys or sells on each trade. Additionally, *FinBuilder* enables these settings to be defined at trader initialisation.

The main differences in behaviour arise from the agents' buying and selling strategies, and from their reaction to market macroeconomic events. For example, being the most primitive, the Simple Agent type is designed to ignore trend indications given by market macroeconomic events, while the Intermediate Agent and Advanced Agent behaviours react to these events. The reasoning behaviour of the agents is implemented through a series of conditional statements of the form

$$(C_1 \& ... \& C_n) \rightarrow do : A_i, ..., A_j$$

for all C conditions and A actions. Agents evaluate each conditional statement to true or false by consulting the financial ontology. The statements vary depending on the sophistication of each trader agent. For example, an agent of intermediate intelligence incorporates the following conditions in its behaviour:

```
(company has loss ) →
        do: suspend trading for x time
( drop in profit) →
        do: suspend trading for x time
(company is being taken over) →
        do: buy shares
(currently hold takeover target company shares) →
        do: suspend trading for x time; sell shares
```

An advantage of the use of ontologies is that the conditions can be classified through description logic reasoning. For a simplistic example using the ontology illustrated in Figure 1, we could create defined concepts GoodInvestmentCompany as a subconcept of the concept Company. The GoodInvestmentCompany is defined as:

$$GoodInvestmentCompany \equiv Company \sqcap \ni hasEvent.TakeOverEvent. \quad (1)$$

This simply means that a GoodInvestmentCompany is Company and is being taken over. We then create another concept called GoodStock as a sub-concept of Stock to define the stocks issued by instances of GoodInvestmentCompany.

$$GoodStock \equiv Stock \sqcap \forall isIssuedBy.GoodInvestmentCompany \tag{2}$$

Through the use of inferencing we could derive instances of the concepts 'GoodStock' which can be used by the trading agents.

Each agent also has a trading portfolio, comprising of realised and unrealised profit tables. These can be viewed at any time during a *FinBuilder* simulation, and are dynamically updated every time an order transaction is successfully completed. The updating of the portfolio is carried out through the passing of information committed to the portfolio ontology layer. In addition, a graph of the profits over the total trading time can be viewed and is updated automatically.

An example of a scenario would start with a macroeconomic event agent creating an instance of TakeOverEvent concept. The instance of the TakeOverEvent concept is shown below by an OWL/RDF representation.

```
<TakeOverEvent rdf:ID="takeOverByCompanyX">
    <isTakenOverBy rdf:resource="#CompanyX"/>
    <eventBelongsTo rdf:resource="#CompanyY"/>
</TakeOverEvent>
```

This is sent through the Macroeconomic Event Layer of the financial ontology. Because the trader agent shares the same ontology, it would immediately understand the concept and compute a response. Depending on the sophistication of the trader agent reasoning, conditional statements will be evaluated using the TakeOverEvent concept. A response by the trader agent will either be nothing or creation of an instance of OrderDetails with attributes representing sell or buy orders of certain quantities of stock. An example of an OrderDetails instance in OWL/RDF is shown below:

```
<OrderDetails rdf:ID="OrderDetailsInstance16">
    <amount rdf:datatype="&xsd;int">10</amount>
    <price rdf:datatype="&xsd;float">143.2</price>
    <orderType rdf:datatype="&xsd;string">Buy</orderType>
    <stockOrder rdf:datatype="&xsd;string">CompanyXStock</stockOrder>
</OrderDetails>
```

The instance of OrderDetails is then sent to the order agent through the order layer of the ontology. Once matching and validation is complete, the instance of OrderDetails get passed onto the order processing agent which in essence updates our virtual stock market.

4.3 Implementation Evaluation

We developed an evaluation strategy based on a heuristic evaluation technique described in [19]. This involved both testing *FinBuilder* with predefined inputs and

demonstrating it to a number of different individuals with varying knowledge and expertise in the fields of information technology and finance.

We ran *FinBuilder* using several variations of macroeconomic events derived from the financial market data. Figure 5 shows the graph of the portfolio values of a Simple Agent and an Intermediate Agent, that both traded with an equal number of shares from the same company over a common time period.

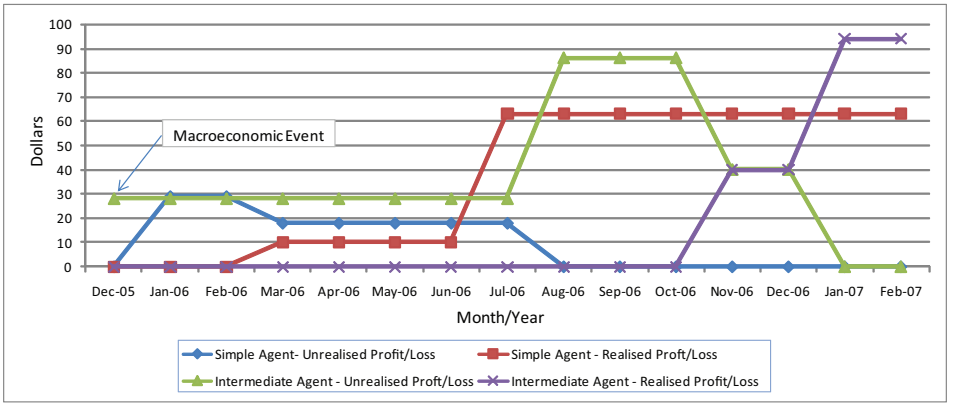


Fig. 5. Portfolio values of a Simple Agent Trader and an Intermediate Agent [18]

The macroeconomic event that occurred during this period was an announcement that the company was the target in a takeover. This announcement occurred at the start of the trading period – in Dec-05.

The Simple Agent, which was not responsive to the macroeconomic event, continued trading, as it normally would have. The black line on the graph goes to zero in Aug-06 as the agent has sold all its holdings, and has realised all its profits. The Intermediate Agent, on the other hand, was receptive to the company announcement through interaction with the financial ontology and reasoning. As a result it ceased trading for a short time to allow for market stabilisation, before re-commencing. In this instance, its strategy was successful. While the results obtained by each simulation were not always the same, they did show that *FinBuilder* successfully demonstrated the use of ontologies with heterogeneous agents within the financial domain.

In addition to studying the *FinBuilder* results, a set of evaluation parameters was defined, under which individuals carried out their evaluation. These parameters focussed on evaluating *FinBuilder* from a user perspective and were: Ease of Use, Flexibility, Scalability, Modularity and Domain Relevance. We asked those familiar with the finance area to provide their evaluation of the first and last parameters, and those with IT expertise to rate all the parameters.

Users of the tool gave the prototype on average 9/10 for relevance to the domain, and useability was given an average rating of 8/10. Those who had IT experience found their main problem was not so much in understanding how to use the interface but rather in understanding the financial concepts well enough to use the interface.

FinBuilder was demonstrated under a Windows platform, however JADE framework is also able to integrate with web browsers and Java Applets, so the application could be translated into a web service in the future, enabling greater flexibility. Similarly, due to the underlying JADE infrastructure, the prototype may be run on multiple computers with little complication. Hence it was assessed as being scalable.

The prototype consisted entirely of the financial ontology layers and agents. Hence its design was modular. In addition, coupling was loose, as agents communicated with each other through sending and receiving of messages that subscribed to the ontology. Thus they did not necessarily need to know the names of the other agents to whom they were sending messages to as generic broadcasting techniques could be employed.

The ontology modelled in Protégé enables sharing across applications and agents. The layered approach taken to the development means that concepts could be specified in separate smaller ontologies and then combined into a larger encompassing ontology. This facilitated the reuse and sharing of the ontology as well as providing a degree of interpretability for the agents.

The ontology plays a crucial role as agent communication is solely carried out through the passing of messages that subscribe to the ontology. This means that whenever an agent receives information for another agent – for example, when the Order Agent receives a sell order from a Trader Agent – no meaning translations are required to understand the communication. Thus, the need for the actual financial domain information to be coded at the infrastructure layer in order for all agents to understand is removed. Also eliminated is the need for human interpretation and supervision to facilitate agent reasoning and dictate behaviour.

5 DISCUSSION

Through the process of developing the *FinBuilder* Tool, we have derived a Design Methodology for Ontology-Based Multi-Agent Applications (MOMA) [23]. MOMA attempts to systematically address the lack of support for ontologies in existing Agent-Oriented Software Engineering (AOSE) methodologies. MOMA addresses some of the difficult problems of ontology development using recent developments, such as Grounded Theory (GT) described next.

Identification of concepts for the purpose of ontology modelling can be a very time-consuming task. It also follows a very implicit and intuitive process. To make it easier for domain experts (who might not have expertise in knowledge engineering), a more methodological approach is needed. Hence, the identification of concepts and relationships is guided by the principles of Grounded Theory (GT) [21, 20]. GT facilitates the production of core categories and relationships from data through

a systematic method of constant comparison where new data is continuously compared to existing data. Although GT originates in the social sciences, it has been proven to be valuable when applied to ontology construction [22]. The key points are marked with a series of codes, which are extracted from the text. The codes are grouped into similar concepts, in order to make them more workable. From these concepts categories are formed. In the context of ontology, the codes are extracted from requirements and domain information. Concepts and sub-concepts are the results.

MOMA involves both knowledge engineering as well as agent development. Hence it is broken down into two parts: ontology development and agent development. Ontology development includes the modelling and representing the ontology as described in Section 3 to the code generation of the ontology. Agent development involves the design and development of the agents and its environment using agent theories. The MOMA process can be summarised as the following sequence of steps for ontology development:

Step 1: Identify Generic, Domain and Task ontology – Gathering of domain knowledge, identifying concepts and building upper layer ontology. Reuse of existing ontology can also be done in this step.

Step 2: Customising Domain ontology for Application – Extending ontological concepts from the previous step for specialised domains and application concepts.

Step 3: Building the Mediation ontology – Mediation ontology is used for heterogeneous multi-agent systems that make use of external entities. It provides a layer of abstraction to those external sources.

Step 4: Building the Communication ontology – The communication ontology will help define the syntax in which the agents communicate with.

Step 5: Adding logic through Rules and Axioms – Although ontology, through languages like OWL, support reasoning, it is very hard to model complicated logic using ontology. This is the reason for the introduction of rule languages such as SWRL (A Semantic Web Rule Language). This step helps define logic that can not be expressed in ontological representation languages such as OWL.

Step 6: Specifying ontological mappings between application ontology – Ontological mapping is a semantic correspondence between two concepts of two different ontology. Mapping in this step is used to bridge the semantic gap between concepts in those application ontologies.

Step 7: Code Generation – This step involves the generation of the ontology into semi-executable code. This code will then be used in the agent implementation.

Table 1 is an evaluation summary of the objectives of MOMA when applied to *FinBuilder* as a case study.

Objective	Evaluation
1. Reuse and sharing	MOMA provides a structured meta-model to model the ontology. This meta-model guides the user into creating ontologies that can be easily shared and reused. This meta-model also allows the reuse of existing generic ontologies.
2. Move business logic and domain knowledge from underlying agent code to higher level	Domain knowledge has been moved from the agent code to the ontology. However, some of the behaviour and business logic of the agents themselves still needs to be coded in the Agent Development Part. This is due to the fact that generation of code for axioms and rules is not supported by current tools.
3. Facilitate the use of tools to accelerate development	MOMA is driven by the use of tools as a part of its processes. The use of tools definitely speeded up the development, especially for time consuming tasks such as concept identification.
4. Reuse of existing ontology	Refer to 1
5. Distinguish roles between domain expert and agent developer	There is a clear distinction between the roles of domain expert and agent developer. The two parts of MOMA separates these roles. However, there is still requirement for the agent developer to request information from the domain expert.
6. Usability by domain expert without the agent developer	This objective has not been satisfied. Without the agent developer, at its current state MOMA cannot produce a working agent application. The ultimate goal is to have the ontology be generated into a code that can be plugged directly into a generic agent framework. There is also an assumption that the domain user understands the basics of knowledge engineering.

Table 1. Evaluation of Objectives of MOMA [23]

6 CONCLUSIONS

This paper has presented an ontology-driven approach for the development of intelligent applications based on multi-agent systems, illustrated through a case study in financial applications. This approach helps domain experts and financial researchers experiment with muti-agent mathematical models without having to know the low-level programming details. Basically, application developers have to focus on ontology development and tools, such as Protg that takes care of the generation of the code that works on multi-agent platforms, such as JADE. Although some financial ontologies have existed in the past, they were not designed to drive agents. Hence

this approach looks promising from the point of view of intelligent application development in finance and other business areas. Therefore, we have attempted to help the adoption of this approach through the development of a new Methodology for Ontology based Multi-agent Applications (MOMA) and associated tools for practical deployment.

During the development of *FinBuilder*, our assessment was that the financial ontology provided a useful way of separating the application infrastructure from financial domain knowledge, thereby enabling agents to communicate more effectively. Its use enabled those with greater knowledge in the domain of finance than that of IT to extend and increase the ontology in complexity in an incremental fashion without requiring greater expertise in low-level languages or technologies. This was substantiated by testing, appraisal of results and evaluation by external individuals.

However, we found that it was necessary to change some agent design aspects, though ontologies we were able to help generate most of the code for domain knowledge, some agent logic and behaviours will still need to be implemented with the agent.

More research is needed to establish this methodology in a practical environment.

Acknowledgments

This research was partially funded by the Australian Cooperative Research Centre (CRC) for Technology Enabled Capital Markets (CMCRC).

REFERENCES

[1] Smith, H. (Director of Strategy, E-Business, CSC Europe): The Role of Ontological Engineering in B2B Net Markets, August 2000, http://ontology.org/main/papers/csc-ont-eng.html.

[2] López de Vergara, J. E.—Villagrá, V. A.—Asensio, J. I.—Berrocal, J.: Ontologies: Giving Semantics to Network Management Models, IEEE Network, Special issue on Network Management, Vol. 17, May/June 2003, No. 3.

[3] Swarm Development Group (SDG): Agent-based modelling resource on the World Wide Web, Introduction to Swarm, Accessed April 2004, http://wiki.swarm.org/wiki/.

[4] Neuberg, L.—Bertels, K.: Heterogeneous Trading Agents, Complexity Journal, pp. 28–35, May 2003.

[5] SFI Artificial Stock Market, Accessed March 2004.

[6] Ambroszkiewicz, S.—Cetnarowicz, K.: On the Concept of Agent in Multi-Robot Environment. Workshop on Radical Agent Concepts (WRAC), 2005, NASA Goddard Space Flight Center, Greenbelt, MD (Washington DC) USA, http://www.santafe.edu/sfi/publications/Bulletins/bulletinFall99/news/stockMarket.html.

[7] Alonso, S.—Bas, J.—Bellido, S.—Contreeas, J.—Benjamins, R.—Gomez, J.: WP10: Case study eBanking D 10.7 Financial Ontology DIP. http://dip.semanticweb.org/documenets/D10-7-Stock-Market-Ontology.pdf.

[8] Zhang, Z.—Zhang, C.—Ong, S. S.: Building an Ontology for Financial Investment. In Proc. Of Intelligent Data Engineering and Automated Learning – IDEAL 2000: Data Mining. Financial Engineering, and Intelligent Agents 19. LNCS 1983, 2000.

[9] Zhang, Z.—Zhang, C.: Agent-Based Hybrid Intelligent Systems, LNAI 2938.

[10] Snoussi, H.—Magnin, L.—Nie, J.: Toward an Ontology-Based Web Data Extraction. The Fifteenth Canadian Conference on Artificial Intelligence AI 2002, BASeWEB Proceedings.

[11] González, E. J.—Hamilton, A. J.—Moreno, L.—Marichal, R. L.—Toledo, J.: Ontologies in a Multi-Agent System for Automated Scheduling, Computing and Informatics. Vol. 23, 2004, No. 2, pp. 157–177.

[12] Fan, J.—Ren, B.—Xiong, L.: Modeling and Management of Ontology-Based Credit Evaluation Meta-Model. IEEE International Conference on Systems, Man and Cybernetics (SMC 04), 2004.

[13] Uschold, M.—King, M.: Towards a Methodology for Building Ontology. In workshop on basic ontological issues in knowledge sharing: International Joint Conference on Artificial Itelligence, pp. 373–380, 1995.

[14] Wooldridge, M.: Reasoning about Rational Agents. The MIT Press Cambridge, Massachussetts/London, England, 2000.

[15] Java Ontology Beangenerator, Accessed July 2004, http://www.swi.psy.uva.nl/usr/aart/beangenerator/.

[16] JADE – Java Agent Development Framework, Accessed July 2008, http://jade.tilab.com/.

[17] Laclavík, M.—Balogh, Z.—Babík, M.—Hluchý, L.: AgentOWL: Semantic Knowledge Model and Agent Architecture, Computing and Informatics, Vol. 25, 2006, pp. 419–437.

[18] Sujanani, A.—Ray, P.—Paramesh, N.—Bhar, R.: The Development of Ontology Driven Multi-Agent Systems: A Case Study in the Financial Services Domain. ACM International Conference Proceeding Series, Vol. 87, Hong Kong, 2005.

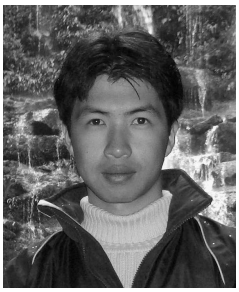
[19] Ray, P.: Integrated Management from E-Business Perspective – Concepts, Architectures and Methodologies. Kluwer Academic/Plenum Publishers, New York, 2003.

[20] Strauss, A.—Corbin, J.:Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. Second edition. Sage Publications, Thousand Oaks 1998.

[21] Strauss, A.—Corbin, J.: Grounded Theory Methodology: An Overview, Handbook of Qualitative Research, Sage Publications, Thousand Oaks, 1994, pp. 273–285.

[22] Kuziemsky, C.—Downing, G.—Black, F.—Lau F.: A Grounded Theory Guided Approach to Palliative Care Systems Design International Journal of Medical Informatics. Vol. 76, Issue null, pp. S141–S148.

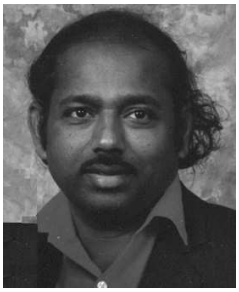
[23] Ying, W.: Design Methodology for Ontology-Based Multi-Agent Applications. Master of Philosophy thesis, School of Information Systems, Technology and Management, University of New South Wales, Sydney, 2009.



Weir Ying is a recent graduate of Bachelor of Software Engineering and Bachelor of Commerce from the University of New South Wales. He is currently undertaking research in the area neural networks and network management. He has interest in the areas of network management, multi-agent systems, Semantic Web technologies and Ontology driven systems.

Anjalee Sujanani recently completed her Masters degree in computer science from Stanford University with a specialization in databases. She obtained her Bachelors in software engineering from the University of New South Wales, Sydney, Australia. Her research interests include data modeling, text retrieval and data mining, software design and implementation, database management systems and strategy in technology.

Pradeep Ray is the Director of the new Asia Pacific ubiquitous Healthcare Research Centre (ApuHC) at the University of New South Wales, Australia. Ontology-Based Multi Agent Systems (OBMAS) is one of the three major research programs at APuHC. His research interests include e-Health, ubiquitous care, networked network/systems/services management, network security, networked enterprise services in various business sectors, such as telecommunications, healthcare and finance. He has more than one hundred international refereed publications (including two research books) in these areas. He has been working for a decade on the use of Internet and related technologies for the healthcare including aged care. He is the founder of IEEE Healthcom conference that brings together people from healthcare, information technology and business to discuss problems and emerging solutions in e-Health that includes aged care using information technologies. He has also lead a number of international initiatives in e-Health, such as the IEEE/ITU-D Mobile e-Health for Developing Countries and the International Ubiquitous Healthcare (u-Health) Initiative. He has been the Chair of the IEEE Technical Committee on Enterprise Networking (EntNet). More details can be found at his home page `http://www.apuhc.org/pradeep`.



N. Paramesh is a Senior Lecturer in the School of Computer Science and Engineering, University of New South Wales, Sydney, Australia. He carries out research in the areas related to agent technology and applications in problem solving in dynamic situations. He is currently involved in the design and implementation of ontology-based agent dialogs in enterprise applications.

Damien Lee is a Ph. D. student in the School of Finance at the University of New South Wales. He holds a Bachelor of Commerce/Bachelor of Science degree majoring in actuarial studies and computer science. His current area of research includes stochastic volatility modelling and forecast density evaluation.

Ramaprasad Bhar completed the Ph. D. in quantitative finance in 1997 from UTS on non-Markovian term structure of interest rate modelling. Prior to joining academia in 1992, he worked in system software development for several years in various capacities in India, Australia, and The Netherlands. He studied computer science at the University of Waterloo, Canada with a scholarship from the Canadian Government. His industry experience includes multinational firms like Credit Lyonnais, Nederland and Unisys, U.S.A. He published two research intensive books with Springer in 2004 and 2005 jointly with S. Hamori, Kobe University, Japan. He was awarded the fellowship of the Japan Society for the Promotion of Science in 2005. His current research interests include stochastic modelling, pricing credit default swap, asset pricing subject to Markov chain and application of copulae to financial market problems. Personal website is at `www.bhar.id.au`.