Agent Services-Oriented Analysis and Design  
---- Building Open Enterprise Infrastructure  
Supporting Trading and Mining  

(Doctoral Assessment)

Longbing Cao

Supervisor: Prof. Chengqi Zhang  
Advisor: Prof. Jiawei Han  
(Uinv. of Illinois at Urbana-Champaign)

Faculty of Information Technology  
University of Technology, Sydney  

July 2004
Contents

Abstract
1 What’s the problem?
1.1 How does the problem emerge?
1.2 What specific problems from financial markets?
1.3 What’s my specific problem?

2 Objectives
2.1 Can trading and mining be supported in one system?
2.2 What is expected for a system supporting both trading and mining?
2.3 What are main research objectives?

3 Related work
3.1 Similar systems
3.1.1 System classification
3.1.2 Similar systems
3.2 Research methods
3.2.1 Objects, components, services and agents
3.2.2 Object-oriented, component-based, service-oriented, agent-oriented

4 Research methodology
4.1 Research methodology in my work
4.2 Agent-oriented methodology
4.3 Agent services-driven approach
4.3.1 The agent services-driven approach
4.3.2 Why agent services-driven approach

5 Key research work
5.1 Goal-oriented organizational modeling
5.2 System function
5.3 System architecture
5.4 Agent service ontology and semantic relationships
5.5 Agent service modelling
5.6 Representation and registry of agent services
5.7 Agent service directory
5.8 Agent service communication
5.9 Agent service transport
5.10 Mediation of agent services
5.11 Discovery of agent services

6 Significance and contributions
6.1 Research value
6.2 Application value

7 Evaluation
7.1 Functional evaluation
7.1.1 As an agent-based system
7.1.2 As a trading/mining support system
7.2 Empirical evaluation

8 Conclusions
8.1 An automated enterprise infrastructure integrating both stock trading and data mining
8.2 Agent services-driven approach as design paradigm for building open agent-based systems
8.3 Agent services-driven enterprise infrastructure supporting trading and mining

9 Future work
10 Timetables
11 Thesis content
12 List of related publications

References
Abstract

Traders and researchers in stock marketing often hold some private trading strategies. Evaluation and optimization of their strategies is a great benefit to them before they take any risk in realistic trading. Data miners test their algorithms again and again before publishing or deploying them. In this work, I’ll investigate that it is possible to build an automated open enterprise infrastructure, which can integrate and support both trading and mining. The main functions of this infrastructure include supports for user personification and financial domain-oriented interaction, online plug and play of algorithms or system components, connectivity with huge real stock data, iterative back-testing, recommendation and integration of trading strategies or mining algorithms, online system customization and re-construction, and add-ons of applications from capital markets.

Agent-oriented methodology is used for system analysis and design. A new approach called agent services-driven approach is proposed for setting up this automated enterprise infrastructure. With this new approach, the following problems are investigated: (1) How to do requirements analysis for the system, (2) What functions are required in this infrastructure, (3) What’s the system infrastructure looked like, (4) What domain specific ontologies and semantic relationships exist, and how to name and register agents and services, (5) How to represent agent services, (6) How to use agent service technology to model the problem, (7) How to organize the agent space, (8) How to mediate requests and response in the agent space, (9) How to locate and transfer services, (10) How to transform (query) agent request in the agent ontology space. Answers to these problems actually build up a new theory for agent services-oriented analysis and design. This approach is compatible with FIPA specifications, and powerful for building open complex software systems.

According to the above investigation, a Java agent services-driven enterprise infrastructure -- F-TRADE (Financial Trading Rules Automated Development and Evaluation) -- has been built. Functional comparison and empirical research show that it is powerful enough for supporting trading and mining in capital markets. F-TRADE is actually evolving to be an online test platform for research and application of multi-agent technology and data mining in stock markets, and linking finance and IT to fulfil IT-Enabled Finance in E-Intelligence Group of UTS and Data Mining Program in CMCRC.

Keywords: open enterprise infrastructure, open complex agent systems, trading system, data mining system, agent-oriented methodology, agent services-driven approach, agent service-oriented analysis and design
1 What’s the problem?

In this section, we’ll introduce what background is for my research, and what specific problems I’ll study for my thesis.

1.1 How does the problem emerge?

I am studying in the E-Intelligence Group [1] at FIT, and also work for Data Mining Program at Capital Markets CRC (CMCRC) [2]. In the CMCRC project, one of the main works is to build some platform for the industry-oriented project required by CMCRC, but it must also support research in both the project and academic work. It is actually quite hard to do both research and industry-oriented development. We finally figured out a solution to kill two birds with one stone.

The solution is, we must do problem-driven rather than interestingness-driven research. All problems come from capital markets and industrial requirements. Then, we filter them out, and abstract some as research problems, and others just for industry. But both of them are related. The linkage is whether we can set up an intelligent infrastructure, and make it support both development requirements from CMCRC, and research for PhD studies. Furthermore, we investigate some other research problems, for instance, data mining in the capital markets, from capital markets through the infrastructure, and feed the results back to improve the infrastructure.

1.2 What are specific problems from financial markets?

So, here we’ll investigate what problems there exist in CMCRC project. I’ll finally figure out a specific problem which has both industry value and research value. It will be the problem I’ll focus on for my PhD studies, and is hoped that it can also benefit both CMCRC program and research work in E-Intelligence group.

In investigation of the CMCRC project, we find out the following requirements and realities.

(1) Stock traders can take less risk if they back-test their trading strategies in house first. Financial researchers often iteratively evaluate their found trading algorithms before publishing them. Data miners train mining algorithms in a loop-closed process with repeating refinement. However, it is very hard for traders and researchers of finance and data mining to build a system by themselves in order to evaluate their strategies. It is also very time-consuming for them to build a system for trading and mining support.

(2) The current evaluation of algorithms is mostly undertaken on data samples. The samples are downloaded from the Internet or generated by data generators. Size and complexities of the abstracted data are far away from what we interact in the real world. As a result, deployment of these algorithms doesn’t bring about what’s expected. It is actually another big difficulty for them to get huge amount of real and distributed stock data for testing.

In light of these two problems, we thought it would be quite helpful if we could build a platform to deal with them, and make the platform available for anyone on the Internet. This original motivation was also proved to be interesting by some requirements from financial industry and academia. CEO of CMCRC encouraged us to build a simple system to help those financial traders and researchers to evaluate their algorithms.

(3) With lot of high new technologies and related interesting research topics available, for instance, the multi-agent technology for building such a system and data mining in financial markets, we can do much more than the above requirements. We can integrate research and applications together in finance.
In light of these realities, we summarized our work as **IT-Enabled Finance**. This work is expected to use IT to assist finance, and study some problems emergent in capital markets. For this big topic, even though we have many things to do, but we’ll investigate the above three requirements first.

### 1.3 What’s my specific problem?

The key issue for combining and figuring out the above three aspects is to **build a comprehensive and powerful infrastructure** first. I took it as my specific research problem.

Over one year ago, we thought about and proposed to set up an automated and enterprise-oriented infrastructure. First, it is not only a system which is specifically designed for evaluation of algorithms. It can provide many other services, and looks like a virtual service provider. It should support evaluation of both trading and mining algorithms. It should also be very flexible and powerful to support both research and development in the Data Mining Program. It must have capabilities to support distributed computing. For instance, it supports remote uploading of algorithms, accessing multiple data sources on Local Area Network (LAN) or even Internet. This is why we called it an **Automated Open Enterprise Infrastructure**.

So, in order to make the above motivations come into effect, I further investigate how to build such an automated enterprise infrastructure, and what it mainly works for. Some main questions include what functions can be provided, what research methods can be used, how to construct it, what key technical problems inside, and who can benefit from it?

The main purpose of this infrastructure is to support trading and mining in capital markets. It is expected that it can enclose some useful facilities for supporting trading and mining activities, including research and applications, in capital markets. This also means that this infrastructure itself will not act as a purely trading system or a data mining system. It looks more like a test platform, which can be personalized and customized for specific applications as trading and mining. This is why we call it as an Infrastructure.

In light of this purpose, my research work is to build an **Automated Open Enterprise Infrastructure Supporting Trading and Mining in Capital Markets**. I called this infrastructure as **F-TRADE (Financial Trading Rules Automated Development and Evaluation)** [3]. The name shows us that F-TRADE is only for supporting trading. However, data mining can be supported in two ways: (1) data mining technology is embedded for optimizing and generating new trading rules, (2) data mining algorithms can be supported in the system. This infrastructure is the key of my PhD thesis topic.

I’ll also discuss it and answer these questions in details in the later sections of this report.

### 2 Objectives

In this section, I’ll discuss my objectives in solving the above problem. The discussion is mainly from research point of view, but industry requirements will also be considered for this is part of a research and development project.

#### 2.1 Can trading and mining be supported in one system?

Before I go ahead to discuss about my objectives in building the proposed infrastructure: F-TRADE, here I first discuss feasibility of the topic. Three key issues are hidden in the research topic. The first is whether and how we can make it **automated**. Second, whether and how can it be **enterprise-oriented**. The final is whether and how it can **support both trading and mining in one system** The first and the second have been mentioned in Section 1.3 from function side. I’ll discuss them in more details in Section 3, 4 and 5. Here I mainly discuss the final item.

There is obvious difference existing between trading systems and mining systems. However, if we put functions related to problem domains aside, we can find that there are actually many commons shared by them. The following are some mutual features or requirements for trading and mining support systems.
Algorithm is the key constituent for either a trading system or a data mining system. Algorithm may take form as a trading strategy in trading system, but as a data mining model in mining system.

Requirements for dataset are basically similar. Fundamental data attributes and time period (intraday or interday) may get involved are similar. Data pre-processing and post-processing are required for both trading and mining.

Human system interaction could be similar. User profiles, interface element types of input and output parameters, and data types of input and output parameters can be categorized into limited groups, and described by some mutual taxonomy and ontology.

System and knowledge management could be unified. By designing some special data structure, information about algorithm model, knowledge involved, information management of algorithm registration and application, running log, etc. can be managed into unified data store.

Software complexities are similar. Flexible, adaptive, open, secure, and user-friendly supports are expected for both of them. These further decide that uniform system architecture and design patterns can be used to build a place for chum mage by trading and mining.

So, it is feasible for us to combine trading and mining into one system.

2.2 What is expected for a system supporting both trading and mining?

It is obvious that my F-TRADE must first satisfy the above mutual features. Furthermore, it must be powerful enough for supporting the following features. These features are proposed for taking it as a comprehensive and generic infrastructure. I would like my infrastructure have the capacity to adaptively support emergent requirements rather than predefined specific tasks. The following are requirement functions I would like this generic system has.

1. It must support integration of heterogeneous and distributed data sources, and transparent to operational systems. For instance, it may access legacy systems remotely in evaluating an algorithm or providing data services.

2. It can support multiform of algorithms both from trading and mining, multiform of data sources, multiform of user types and profiles.

3. Algorithms, system modules, user information, information and knowledge resource can be plugged into and removed from the system locally and remotely by different people.

4. After plug in of algorithms or other components, they can be automatically registered into the system. Another instance, the user interfaces can be generated automatically. That means it support software plug and play.

5. Variant user profiles and financial domain concepts can be supported in human system interaction.

6. It is expandable for future finance-oriented applications, for instance, investment analysis and decision support, e-commerce pattern for system marketing.

7. Privacy of plugged algorithms can be kept.

In addition, as a trading and mining support, it must also support optimization of parameters, presentation of outputs, online extraction and transformation of data from remote data sources, and so forth.

Therefore, this infrastructure must be a practical web-based workspace living in a real open environment. It must be a distributed problem solving system. These are main considerations we call it as automated and enterprise-oriented.

2.3 What are main research objectives?

Based on the above expectation of the infrastructure, and the requirement for PhD thesis, I outline the following as my main research objectives.
(1) Investigating what kind of concrete functions can be available in the automated enterprise infrastructure, so that simulated trading and mining can be flexibly supported on it.

(2) What kind of research methodology and methods are required for building such an infrastructure? Some new approach may be proposed for solving this specific problem.

(3) For the proposed approach, what are key research problems, and what research values are there? How to solve these problems?

(4) Supporting both research and development in academic and industrial projects in CMCRC project, that means it can support and link all work of the PhD students in the Data Mining Program, and evolve into an increasingly powerful workspace.

(5) Some high quality papers and PhD thesis can be generated from this research work.

With respect to (1), we’ll discuss in Section 5.1. Question (2) will be answered in Section 4. Section 5 focuses on discussing problems related to (3). Items (4) and (5) will be touched in Section 6 and 11.

3 Related work

The existing work related to the above objectives can be discussed from the following aspects. (1) From system implementation, what similar systems exist, and what difference between them and mine? (2) With regard to research methodology and methods, what work has been done, and what’s the new requirement for my proposed infrastructure?

3.1 Similar systems

In this Section, I’ll first present a system classification method to categorize all trading and mining systems available, so that the position of my infrastructure can be made clear. Then, I’ll discuss what similar systems are available both from industry and academia, and the main differences between the proposed F-TRADE and its counterparts.

3.1.1 System classification

Both trading systems and mining systems can be classified according to content and its degree disclosed to user system interaction. Basically, we classify both trading systems and data mining into four categories: Black Boxes, White Boxes, Grey Boxes and Glass Boxes.

Black Boxes

A Black Box system is specified entirely in terms of its function and interface. The user has no knowledge of its internal structure or method of implementation, and hence all the knowledge needed to use it must be made explicit by the definition of the system itself and the environment in which it is to be used. The system is used exactly in the form in which it is provided. In human computer interaction, only the externally visible behaviour of a black box is considered and not its implementation or “inner workings”.

Most of commercial trading and mining systems are in Black Box. Black Boxes are generally described as ‘trading systems’ or ‘mining systems’ because they run on proprietary algorithms that are not disclosed. They run these algorithms over database and give users trading/mining recommendations or suggestions, but usually not tell users why. The provider of the system is responsible for the maintenance and quality of the algorithm and system components.

White Boxes

A White Box system is provided with all the source codes so that all details of the structure and implementation of the component can be seen by the user. A White Box system looks more like a Toolbox
system. The system can be modified and adapted to suit the exact needs of the user. This is probably the most widespread type of reuse which operates in an unstructured and ad hoc way.

A Toolbox is a box for holding hand tools. To be classified as a toolbox, the software must either fully disclose algorithms, or let users to decide what the box does by calling tools available in the box. The user will be entirely responsible for the modifications, deployment, quality and maintenance of the system. Usually, this type of system does not create trading suggestions. The user does the analysis and makes his or her own decisions.

Glass Boxes

A Glass Box is a White Box component which is used unmodified. Using a system this way can bring most of the benefits of using Black Box without the disadvantages. All of the knowledge needed to make successful use of the component is not actually made explicit in the interface definition. By examining the internal structure and implementation, the user can glean additional knowledge that was not made explicit in the formal definition. This knowledge can help the user to trade or mine more flexibly and consciously.

Grey Boxes

A White Box could take the form as Grey Box if with only minor modifications. Grey Box is a box staying in intermediate position between Black Box and White Box with some degree of customization by users. Trading/mining systems in Grey Boxes can generate suggestions from proprietary algorithms. However, they provide a general idea of how the formula works and sometimes allow the user to modify the settings or parameters. They may also have an associated tool box component.

For the proposed F-TRADE, it is positioned as a generic and flexible infrastructure. As an infrastructure, it can be a Glass Box or a Grey Box. However, it can be customized as a Black Box, for instance, a trading system for retailers. It may also be personalized as a White Box, for instance, a system framework provided to data miners for them to plug in algorithms and evaluate them by themselves. We define functions and features for F-TRADE so that it can be flexibly positioned as any of the above four boxes.

3.1.2 Similar systems

First, there is no system I can find which can support both trading and mining as we planned in F-TRADE. However, there are many computerized trading systems [8] and data mining systems [9] available, respectively.

Computerized trading systems [8] are being aggressively marketed in both Australia and the world. There are probably more than 100 of such kind systems. In Australia, there are more than 30 trading systems sold in the markets [10]. The most similar trading system to the proposed FTRADE is TradeStation [11]. TradeStation® is a commercial trading system, which claims to be a complete environment for designing, testing and optimizing trading systems. Table 1 in Section 7 further lists the comparison of these two systems from three aspects like trading supports, data supports, and system supports. The capacities in data and system supports are the main concern in the research work of the thesis. The strength in F-TRADE has been described as in Section 2.2. The main difference will also be clarified in Section 7.1 as functional performance evaluation.

Data mining [12] is getting more and more popular in research, deployment of data mining in real world is increasing. There are several commercial data mining systems available, for instance, IBM Intelligent Miner [13], SAS Enterprise Miner [14], SPSS Clementine [15], Angoss [16], and open source WEKA [17]. In this work, my motivation of making this infrastructure to support data mining can be explicitly defined as follows.

(1) The main objective is not to develop comprehensive mining algorithms, or make it as a standalone mining system; it is expected to be a mining support platform, which provides capacities for evaluating mining algorithms. These capacities include plug-in support, interface automatic generation, data source configuration and management, output evaluation.

(2) The data mining research in capital markets can be supported to be undertaken on this platform. The research work could be in form of self-existent algorithms, or be integrated into trading strategies.
(3) Capital data crossing markets can be online extracted and prepared for data mining on requests.
(4) It could be evolved into a data mining system if required.

3.2 Research methods

Research methods which can be used for developing an infrastructure like FTRADE include Object-Oriented (OO) [18] methodology, Component-Based (CB) [19] methodology, Service-Oriented [20] methodology, and Agent-Oriented (AO) [21] methodology. Before I discuss these methodologies, the key computational entities in these methodologies will be discussed first.

3.2.1 Objects, components, services and agents

Objects [18], Components [19], Services [20] and Agents [21] are some very popular higher level of abstraction models, which are currently used by software academic designers and industrial architects to construct software systems in modelling the real world.

Objects

Objects are defined as computational entities that encapsulate some state, are able to perform actions, or methods on this state, and communicate by message passing.

In implementation, the “Class” is the design-time component that developer works with. "Object" is an instance of the class, or in other words the "Class" is a template for the "Objects".

We may define an Object as a computational entity, which
- store data and those operations (methods) that manipulate the data
- is a unit of instantiation, has a unique identity
- may have state and this can be externally observable
- encapsulates its state and behaviour

Components

The terms Components have become overloaded during the past few years. The following definitions can be found from academia and industry.

- A component is a self-contained, clearly identifiable, physically realizable, predefined entity that provides a well-understood function and is intended to be assembled, using a well-defined architecture and interfaces, with other components to provide more complex functions [22].

- A component is a unit of packaging, distribution, or delivery that provides services within a data integrity or encapsulation boundary [23], or

- A component is a language neutral, independently implemented package of software services, delivered in an encapsulated and replaceable container, accessed via one or more published interfaces. While a component may have the ability to modify a database, it should not be expected to maintain state information. A component is not platform-constrained nor is it application-bound [24].

Services

A service is a function that is well-defined, self-contained, and does not depend on the context or state of other services. Software services are discrete units of application logic that expose message-based interfaces suitable for being accessed across a network. Service-based architectures [20] permit very flexible deployment strategies; rather than requiring that all data and logic be resident on a single computer. The service model allows applications to leverage networked computational resources. An application architecture that depends on interoperable services can provide high-value business logic and state management.
Although Web Services [25] are the most popular way of implementing services, but a service doesn't have to be a "Web Service", and you can have a Web Service which is not a service. Many people use web services as a transport protocol so we shouldn't "Services" and "Web Services" interchangeably.

In web services, Services are autonomous platform-independent computational elements that can be described, published, discovered, orchestrated and programmed using XML artifacts for the purpose of developing massively distributed interoperable applications.

**Objects, components and services**

Basically, we can take components as “natural extension” of objects. An Object is a design time artefact for the developer to work with before compilation into one or more (or less) components. It is what the developer works with and understands. A component is an assembly, which is made up of however many objects. What makes the collection of objects a component? Because they are in the same assembly! In implementation, a component is the actual DLL or exe instantiated in the system at run time and executing the designed functionality. It is what the machine works with and understands.

If you are debugging functionality you work with objects, if you are tuning systems or into locking and transactions then you work with components.

A service is a product that cannot be understood as either a component or object: it is more about activity than about structure. Their platform neutral and self-describing nature and particularly their ability to enable business collaborations provide significant competitive advantages.

**Agents**

An agent is an encapsulated computer system situated in some environment and capable of flexible, autonomous action in that environment in order to meet its design objectives [26]. This definition means that Agents are:

- clearly identifiable problem-solving entities with well-defined boundaries and interfaces;
- situated (embedded) in a particular environment over which they have partial control and observability—they receive inputs related to the state of their environment through sensors and they act on the environment through effectors;
- designed to fulfill a specific role—they have particular objectives to achieve;
- autonomous—they have control both over their internal state and over their own behavior;
- capable of exhibiting flexible problem-solving behavior in pursuit of their design objectives—being both reactive (able to respond in a timely fashion to changes that occur in their environment) and proactive (able to opportunistically adopt goals and take the initiative) [27].

Compared with other computational entities, Agents are computer system with two important capabilities. First, they are at least to some extent capable of autonomous action – of deciding for themselves what they need to do in order to satisfy their design objectives. Second, they are capable of interacting with other agents – not simply by exchanging data, but by engaging in analogous of the kind of social activity that we all engage in every day of our lives: cooperation, coordination, negotiation and the like.

**Agents, objects and components**

The main distinctions existing between “traditional” view of object and agent are as follows.

A “traditional” object can be thought of as exhibiting autonomy over its state: it has control over it. But an object does not exhibit control over its behaviour. However, agents embody a stronger notion of autonomy than objects. For an object, the decision lies with the object that invokes the method. While for an agent, the decision lies with the agent that receives the request. This distinction has been summarized as “Objects do it for free; agents do it because they want to”.

Second, “traditional” objects cannot deal with flexible behaviours; agents are capable of integrating reactive, proactive and social behaviours into one system.
The third distinction is that there is only a single thread of control in an object-oriented system; while in a multi-agent system, each agent is assumed to have at least one thread of control, the system is inherently multi-threaded.

Even though the distinction between the contemporary view of objects and agents is becoming less sharp with time, for instance, some researchers proposed component-based design of intelligent agents [28], it is clearer that agent-based abstractions suit when shifting the attention from object/components to complex systems [29].

3.2.2 Object-oriented, component-based, service-oriented, agent-oriented

With the emergence of objects, components, services and agents, corresponding software engineering methodologies are proposed and under studies at the same time. They are object-oriented methodology, component-oriented methodology, service-oriented methodology and agent-oriented methodology.

Object-oriented methodology

Object-oriented analysis and design [18, 30] model the world in terms of objects that have properties and behaviour, and events that trigger operations that change the state of the objects. Objects interact formally with other objects [30].

In object-oriented programming, a program execution is regarded as a physical model, simulating the behaviour of either a real or imaginary part of the world. It can basically be expressed in the following formula:

\[
\text{ObjectOrientedProgramming} = \text{Polymorphism} + (\text{Some}) \text{ Late Binding} + (\text{Some}) \text{ Encapsulation} + \text{Inheritance}
\]

Component-based methodology

The component-based approach [19] encourages the creation of reusable components and the production of new software systems through the assembly of preexisting components. There are two engineering drivers in the development of a component-based system:

- **Reuse.** The ability to reuse existing components to create a more complex system.
- **Evolution.** By creating a system that is highly componentized, the system is easier to maintain. In a well-designed system, the changes will be localized, and the changes can be made to the system with little or no effect on the remaining components.

In component-based programming, the main elements can be expressed in the following formula:

\[
\text{ComponentBasedProgramming} = \text{Polymorphism} + (\text{Really}) \text{ Late Binding} + (\text{Real, Enforced}) \text{ Encapsulation} + \text{Interface Inheritance} + \text{Binary Reuse}
\]

Many CB developments are OO. The possible tension between the two approaches arises because CB aims to encapsulate function in large components that have loose couplings. OO is also about encapsulation, but with objects, and there tends to be highly complex coupling between objects.

Object-oriented development and component-based design are both based on the following hypothesis and belief: complex systems can be built from smaller components, bound together by software that creates the unique behavior and forms of the system. However, the practical requirements from dynamic and flexible real world cannot be satisfied with software based on the above software engineering methodology. The main reasons come from the following software complexities: a software system with autonomy and flexibility. These lead to the emergence of agent-oriented methodology [21, 31].

Service-oriented methodology

Service-oriented methodology [20] is the new emerging paradigm for distributed computing and e-business processing. It has evolved from object-oriented and component computing to enable building agile networks of collaborating business applications distributed within and across organizational boundaries.
A service-oriented architecture is essentially a collection of services. These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed.

Combined with recent developments in the area of distributed systems, workflow management systems, business protocols and languages, services can provide the automated support needed for e-business integration both at the data and business logic level. They also provide a sound support framework for developing complex business transaction sequences and business collaboration applications.

The application of the service-oriented computing to web resources is manifested by Web services to provide a loosely coupled model for distributed processing.

Agent-oriented methodology

Agents may be used by software developers to more naturally understand, model, and develop an important class of complex distributed systems. However, there is a fundamental mismatch between the concepts used by object-oriented developers and the agent-oriented view [32, 33]. In particular, extant approaches including OO and CB fail to adequately capture an agent's flexible, autonomous problem-solving behavior, the richness of an agent's interactions, and the complexity of an agent system's organizational structures [34]. For these reasons, agent-oriented methodology is introduced specially tailored to the analysis and design of agent-based systems. It will be introduced in details in next section.

4 Research methodology

Based on related work analysis and literature survey in Section 3, I choose the agent-oriented methodology as computing paradigm for system analysis and design. In agent-oriented methodology, I’ll investigate a new research direction – agent services-driven approach for building multi-agent systems. There are many open problems existing in agent services-driven approach. I’ll discuss what main research problems are. In section 5, I’ll discuss my strategies for solving these problems.

4.1 Research methodology in my work

From the above analysis and comparison of these computational entities – objects, components, services and agents, we get to know that Services and Agents are more suitable for modelling high-value business logic, state management, automated and flexible functional elements in a complex software system. Since the strengths and benefits of agents and services are different. Services are discrete units of application logic that expose message-based interfaces suitable for being accessed across a network; while agents are suited to model entities with autonomous actions and cooperative ability to archive their design objectives. Therefore, it is powerful for modelling some particular problems if we combine functionalities of agents and services together. This is the original thoughts about building Agent Service-Driven Approach. The objective of this approach is to build a systematic and practical theory for agent service-oriented analysis and design.

In my work, the agent-oriented methodology will be used for system analysis and design; services of agents will be investigated for modelling business logics, organizing and managing interoperation in an enterprise computing, and wrapping legacy applications and the like. So, I formally call this method as Agent Services-Driven (ASD) approach in the multi-agent field; the software engineering is called agent service-oriented analysis and design.

As a new agent-based approach, I’ll investigate the basic rationale of agent services-oriented technology. Further work will be on agent services-oriented modelling, analysis and design.

4.2 Agent-oriented methodology
Agent-oriented approach to software engineering means decomposing the problem into multiple, autonomous components that can act and interact in flexible ways to achieve their set objectives. The key abstraction models that define the agent-oriented mindset are agents, interactions, and organizations. Explicit structures and mechanisms are often used to describe and manage the complex and changing web of organizational relationships that exist between the agents.

The argument in favor of an agent-oriented approach to software engineering includes [35]:

- Show that the key abstractions of the agent-oriented mindset are a natural means of modeling complex systems;
- Show that agent-oriented decompositions are an effective way of partitioning the problem space of a complex system; and
- Show that the agent-oriented philosophy for modeling and managing organizational relationships is appropriate for dealing with the dependencies and interactions that exist in complex systems.

Research in the area of agent-oriented software engineering has expanded significantly in the past few years. Several groups have started addressing the problem of modeling agent systems with appropriate abstractions and defining methodologies for MASs development. Shehory and Sturm [36] gave a survey of this topic. Several complete methodologies, for instance, MASE [37], MESSAGE [38], TROPOS [39] and Gaia [34], for the analysis and design of multi-agent systems (MASs) have been proposed. The following discuss about some of these agent-oriented methodologies.

MASE

In analysis of the MASE Methodology [37], the requirements are used to define use-cases and application goals and sub-goals, and eventually to identify the roles to be played by the agents and their interactions. In design, agent classes and agent interaction protocols are derived from the outcome of the analysis phase, leading to a complete architecture of the system. MASE is only suitable for closed agent systems.

MESSAGE

The MESSAGE methodology [38] defines an organization in terms of a structure, determining the roles to be played by the agents and their topological relations (i.e., the interactions occurring among them). In MESSAGE, an organization is also characterized by a control entity and by a workflow structure. They determine the laws that agents’ actions and interactions have to conform to during execution.

TROPOS

The TROPOS methodology, first proposed in Bresciani et al. in 2001 and refined in Kolp et al. [39], recognizes that the organizational structure is a primary dimension for the development of agent systems, and that an appropriate choice of it is needed to meet both functional and nonfunctional requirements. TROPOS methodology defines uniform and coherent guidelines for the activities of both early and late requirements engineering.

Gaia

Gaia methodology, proposed by Wooldridge et al. [34] and refined by Franco et al.[40], is founded on the view of a multi-agent system as a computational organization consisting of various interacting roles. It provides a clean separation between the analysis and design phases. It defines a fully elaborated role model, together with an accurate description of the protocols in which the roles will be involved, overall organizational structure, global organizational rules and the modeling of the environment.

The Gaia methodology is both general, in that it is applicable to a wide range of multi-agent systems, and comprehensive, in that it deals with both the macro-level (societal) and the micro-level (agent) aspects of systems. It exploits the organizational abstractions to provide clear guidelines for the analysis and design of complex and open software systems. According to what features are expected for F-TRADE in Section 2.2, it is an open, flexible and automatic infrastructure, and needs to support capacities like privacy-keeping, Adaptability and interoperability. So, in this work, I’ll use Gaia methodology to do system analysis and design of F-TRADE.
4.3 Agent services-driven approach

Agent-oriented methodology encloses some intrinsic strength for dealing with open enterprise complex software system like F-TRADE. As discussion shown in the last section, this does not indicate that there are some standard or commonly recognized development approach available for solving any specific problems. In reality, it is the work of every research group in multi-agent field to develop suitable agent-based approach in terms of its own particular requirements and agent-oriented methodology. Of course, we should absorb and extend on top of any contributions available in this research field.

4.3.1 Why agent services-driven approach

As I have discussed in Section 1.3, 2.2 and 2.3, an open enterprise infrastructure like F-TRADE is very complicated. It is not a simple complex system. It could be a very large open agent-based system. Here Very Large means the number of sub-systems and sub-system components in the system could be very big; there are multiple levels existing in the system; the business organizations and users that the system gets involved could be very big on the Internet.

For these considerations, most of existing agent-based approaches cannot handle the development of F-TRADE very well. The main challenges come from the following software complexities. (1) It is an open system. However, most of the current agent-based technologies under investigation are actually only suitable for closed or semi-open system. (2) It must be interoperable enterprise applications-oriented. This brings new open problems for “traditional” agent-based architecture. For instance, how to locate, transport and interoperate with those distributed heterogeneous enterprise applications. (3) It lodges in Internet-based environment. Most of agent systems are standalone systems or at most local network-based. F-TRADE is designed to be Internet-oriented. In this environment, new technology is required to locate, transport and interoperate services and applications on the Internet.

These problems are mainly open and enterprise-oriented, so computational entities with interoperable, flexible and automated capabilities are needed in a network-based computing environment. To the above end, some new technology in agent field must be investigated. As a newly emergent research direction, agent service technology [41, 42, 43] is very promising for building open enterprise automated systems. Here I further promote it as an Agent Services-Driven Approach. I’ll discuss its rationale and potential benefits for building an open enterprise agent system such like F-TRADE.

4.3.2 The agent services-driven approach

The Agent Services-Driven approach is a new research direction. Some similar work can be found from FIPA Abstract Architecture [41], Java Agent Services [42], Web Services [44] and Service-Oriented Architecture [20]. These technologies either focus on specific services like web services and Java agent service, or have no close and explicit relations to agent technology like service-oriented architecture. Different from them, Agent Services-Driven approach will be investigated as a generic agent-based approach. It is not necessary for it to be in Java or in Web Services; this is similar to studies on Service-Oriented Architecture, it is not necessary for SOA to be web services-based.

FIPA Abstract Architecture is proposed by FIPA, it defines (1) A specification that defines architectural elements and their relationships (this document); (2) Guidelines for the specification of agent systems in terms of particular software and communications technologies (Guidelines for Instantiation); (3) Specifications governing the interoperability and conformance of agents and agent systems (Interoperability Guidelines). In this specification, abstract agent model and service model are described respectively. My work will be based on the abstraction of FIPA agents and services models, and keeps compatible with other FIPA specifications related to agents and services. Furthermore, as a complete and operable agent-based approach, Agent Services-Driven approach will investigate agent services-based system modeling, analysis and design concretely and systematically. This needs (1) to consider more concretely and implementation-oriented, (2) extend or complement FIPA specifications on demand. So, from this aspect, this Agent Services-Driven approach is instantiation and concrete extension of FIPA Abstract Architecture specifications and other agent related specifications.
In Agent Services-Driven approach, a system and its environment will be explicitly abstracted and modelled in the metaphor of organization and society [45]. Activities of both early and late requirements engineering, roles, organizational hierarchy and structure, local and global organizational rules, interactions inside of the organization and with external humans and applications, are the main abstraction work. In this work, the abstraction and modelling technique Gaia [40] will be used.

Since Gaia is still under development in supporting open agent systems, strengths and functionalities from other related organizational abstractions will also be considered in the modelling and abstraction of F-TRADE. For instance, Agent UML [46] will be used for modelling; goal-oriented analysis [47, 48] will be conducted in early requirement engineering; service-oriented architecture and service abstraction will be explicitly added into Gaia.

In agent service abstraction, a system will be explicitly modelled in basic item atoms as both agents and services in terms of entities and their business logics, respectively. Agents are suited to model entities with autonomous actions and cooperative ability to archive their design objectives. While Services are discrete units of application logic related to the entities, which expose message-based interfaces suitable for being positioned, accessed, transported and discovered across a network. In the design phase, both agent model and its service model will be built.

Before the agent service oriented computing paradigm becomes reality, there is a number of challenging issues that need to be addressed. These include agent service modeling, agent service analysis and design methodologies, organizational architecture with service support, agent service interaction model, development, deployment and composition of agent services, programming and evolution of services and their supporting technologies. All these issues will be covered in the process of modeling, analysis and design of agent services-based system.

The following aspects will be investigated in details in the proposed agent services-driven approach: (1) agent service ontology and representation, (2) naming and registry of agent services, (3) modeling of agent services, (4) agent service directory, (5) agent service transport, (6) agent service mediation and management, (7) agent service discovery. In addition, system functions and agent service-based architecture of the proposed F-TRADE will also be studies.

5 Key research work

Up to now, it is very clear that my research work will be on setting up concrete and complete agent services-driven approach, and how to use agent services-driven approach for building an automated enterprise infrastructure supporting trading and mining in capital markets. The following are some key research work in building an agent services-driven automated enterprise infrastructure like F-TRADE. (1) How to do requirements analysis for agent service-based system, (2) What functions are required in this infrastructure, (3) What’s the system infrastructure looked like, (4) What domain specific ontologies and semantic relationships exist, (5) How to represent agent services, and how to name and register agents and services, (6) How to use agent service technology to model the problem, (7) How to organize and position agents and services in the agent space, (8) How to mediate and manage requests and response in the agent space, (9) How to locate and transfer services, (10) How to transform, query and discover agent service in the agent ontology space. Answers to these questions can actually make abstraction and work process of agent services-driven approach for building automated enterprise infrastructure clearly defined.

5.1 Goal-oriented organizational modeling

As we know, an agent is a goal-driven entity with autonomy and self control of capabilities. Goal-oriented analysis is compatible with the motivation of agent-oriented methodology. The goal-oriented requirement analysis will cover not only functional requirements and nonfunctional requirements, which also be called as early and late requirement analyses.
The main research problem for this work is to extend goal-oriented modeling to agent-oriented modeling. Goal-oriented organizational modeling will be investigated. Integrative modeling which integrates visual modeling and formal modeling will be studies for modeling the agent organizations.

5.2 System function

As a fundamental infrastructure supporting research and development in financial markets, FTRADE presents powerful services. It can be called a virtual service provider, where data services, algorithm services, and system services can be provided [3, 49, 50].

(1) Data services support. Data sources with special formats, for instance, data with format of ODBC, JDBC, OLEDB, FAV (a special format designed for SMARTS [51]), or flat files, can be plugged into and accessed from FTRADE remotely. The FTRADE also provides data access gateway agents for supporting route to remote stock data warehouse. Private data sources held by users on their local host can also be routed if only a public interaction is available. Special data processing agents are designed for extraction, pre-processing and post-processing of data required by a trading rule or mining algorithm.

(2) Algorithm services support. In order to support Plug and Play (Plug-In) and iterative evaluation of algorithms, the following services are required. It supports registration of algorithms user friendly, and automatically identifies and generates personalized input and output interfaces according to the types and customized information both of algorithms and their parameters. Testing data sources, algorithm information, and execution results of algorithms can be easily and flexibly linked and managed with online definition and configuration of database or flat files. Users or providers of algorithms can iteratively change input parameters’ value and evaluate their strategies. Results of algorithm execution can be represented and evaluated in multiple forms including summary report, all simulated records in details, and visualization with point details.

Some advanced supports for algorithms can also be available. Optimal value combinations of input parameters can be mined and recommended to users using data mining technology. Some evaluation metrics and benchmarks can be available. Users can even integrate multiple trading rules to select optimal rules or generate combinatorial signals with data mining-enabled algorithm integration technology.

(3) System services support. A number of functions and services can be available from FTRADE as a trading and mining support infrastructure. User profile and privacy (including user personal information, and the privacy of registered or subscribed algorithms, their results and data sources) can be personalized and protected. With internal supports for design patterns and message passing of agents and agent organization, and global naming, representation, location, configuration, mediation and management of agents, additional system modules and application-oriented components can be plugged into and positioned in FTRADE very effectively. All functional modules in FTRADE can be flexibly customized and re-organized to be a personalized system according to users’ interestness and requirements. Information and knowledge of user agents, algorithm agents, system functional agents are managed in database, algorithm base and knowledge base.

5.3 System architecture

The system architecture of FTRADE [3, 49, 50] is shown in Figure 1. There are five function Centers in FTRADE [3]. They are Administration Center, Algorithms Center, Control Center, Services Center, and User Center.

Direct application components of financial trading and mining can be agentized in FTRADE. These components must and can easily follow specifications of organizational framework and design patterns, and implement particular API for corresponding functional agents. The agents are plugged into FTRADE through Control Center, and specified by naming, activity, location and directory in the organization, and relationship to other agents. Interfaces for user agent interaction are generated according to user profile and agent ontology.
Six types of users including algorithm providers and subscribers can interact with corresponding user interface agents lodging in the above Centers. User requests will be mediated and directed to target functional agents by mediator agents. The target agents will act on the requests with cooperation from other functional agents and services as required, or deliver requests to other relevant agents for further activities.

Heterogeneous data resources, including both local and remote, and multiform data formats, are managed by Data Resource Gateways. Data Source Interface Gateway agents deal with registration and maintenance of data source drivers, link information, and data. Requests for data extraction, preprocessing, transportation and operations on data sources are mediated by Data Source Operation Gateway agents, further actions are taken by functional agents as required.

System database, algorithm base, and knowledge base of F-TRADE are maintained by different types of System Resource Gateway agents according to the types of information and storage. Particular agent services are designed to operate and manage User profiles, ontology of agents, and configuration information of organizational framework and user permission.

![Architecture of F-TRADE](image)

**Figure 1. Architecture of F-TRADE**

### 5.4 Agent service ontology and semantic relationships

It is necessary to abstract the domain specific ontologies [53] in order to understand the problem domain (capital market and its behaviours) and set up corresponding supports for the problem in the system. In addition, as a complex agent system, it is also essential to extract agent and service ontologies [54], so that management and operations of agent services can be handled systematically.

This section will focus on studying of agent service ontologies. Ontologies both of the problem domain -- the capital market and of the agent service-based problem solving system will be abstracted and observed.

On the other hand, understanding and maintenance of semantic relationships [55, 56] between ontologies is also very important for efficient and effective operations of ontologies. Semantic relationships are also fundamental for transformation, looking up, and discovery of ontologies. So, this section will also investigate semantic relationships between the domain specific ontologies and also between problem-solving ontologies.

Ontology transformation [57] will further be looked at with the explicit definition of the semantic relationships. Some transformation rules will be discussed in terms of different types of combinations of agent ontologies.

### 5.5 Agent service modelling
With the understanding of the problem domain, conceptual model and agent services-based system model will be discussed. Conceptual model [58] will be constructed in terms of the system objectives; use case [59] and Agent UML [46] will be used for the modeling. Gaia [40] will be used for the system analysis and design. The following models will be studied: Role Model, Interaction Model, Environment Model, Organizational Rules, Organizational Structure, Agent Model, Service Model, and Agent Service-oriented Architecture.

These models will be built in either period of system analysis or period of system design. Finally, the agent service-oriented architecture will be established as the system architecture of F-TRADE.

5.6 Representation and registry of agent services

Description of agents and services will be studied. Namespace [60] and service root [41] of the agent service-based system will be designed, so that agents and services can be named in a systematic specification. These are work of representation of agent services.

With the namespace and naming mechanism of agent services, specification for registration of agents and services must be studied uniformly in the agent space. Correspondingly, strategies for representation and registration management will be discussed, so that representation and registry of agent services can be handled efficiently in both micro-level (as an agent or a service) and macro-level (as a sub-system or an isolated functional component). That means an agent or a service, for instance, a third party algorithm wrapped as an agent, can be described and registered normally. A sub-system or an isolated functional component, for instance, a financial analysis sub-system externally available wrapped as a big agent, or a third party data source wrapped as a functional component, can also be described and registered in the same way. This function is essential for implementing software Plug and Play.

5.7 Agent service directory

The agent service directory involves agent directory service [41, 42] and service directory service [41, 42].

The basic role of the agent directory service is to provide a location where agents register directory entries. The agent registers with the directory service passing to it a directory entry. Other agents can search the directory-entries to find agents with which they wish to interact.

The basic role of the service directory service is to provide a consistent means by which agents and services can discover services. So that services can register their service descriptions as service directory entries. Also, agents and services can search the service directory service to locate services appropriate to their needs.

The specification of directory service will be investigated. This includes what attributes can be selected into directory entry for positioning agents, and how to describe the directory entry.

The position of agent service directory in the whole agent service-oriented abstraction will also be discussed. This will link some related abstraction elements in the agent service-oriented organizational abstraction.

5.8 Agent service communication

Agent service communication deals with communication model, communicative act, and communication control in agent and service communication.

Message-based communication model will be built. In this model, message structure will be discussed in terms of related message parameters; this can be called as agent message model.

Agent service message model [41, 63, 64, 65] is essential for message packing and delivery before discussing about transport of message. This involves what additional properties will be abstracted and packed into a message body. Besides, it is required to find some expression method to describe message
content. This could be a kind of communication language. For the expression of agent messages, agent ontology will also be involved.

Before the delivery of agent service message, some other communication setting like communicative act, conversation partners’ setting, conversation type and so forth must be defined.

5.9 Agent service transport

Agent service transport mainly deals with representation and transport of messages between agents, and what transport protocol can be used for the delivery of the message. Message transport [41, 61, 62] supports the sending and receiving of messages between agents. Transport type deals with message can be delivered in what type of communication protocol across networks.

In the delivery of messages between agents, messages will usually be transformed into or wrapped as some secure packet probably with transport address and verification information. This message packing will be further embedded into the agent message model.

Agent service communication channel will be set up between agents and services. In the message-based communication, the following issues as transport interfaces, delivering (sending and receiving) and handling of messages, quality and exception handling of communication will be studied.

The specification of transport service will be investigated in details. This includes what attributes can be selected to describe the transport service, and how to represent the transport service of agent.

5.10 Mediation of agent services

The mediation [49, 66] and management [67] of agent services is very important for proper work of large agent system. With the organizational metaphor, an agent space can be viewed as consisting of multiple organizational branches, and there are probably multiple organizational hierarchies existing in the agent society. For instance, these organizational branches could be some relatively isolated functional centers or activity centers, where relevant agents and services work together to fulfill actions or support services in these centers.

So, the mediation of agent services in the agent society could be in two levels. One is to support local meditation in a sub-system; another is to provide global mediation in the agent society. For these objectives, some multi-tier mediation strategies must be designed to deal with management in different levels. Mediation protocols must be considered in the negotiation, mediation logic must be clarified from the ask request to the reply response.

5.11 Discovery of agent services

The discovery [41, 68, 69] of agent services could be in two forms: the first may be to query an agent with which to communicate; the second may be to search for a message.

For the former, an agent may search another agent for some requested actions or information advertised by it. This can be located by agent/service directory service. The request queried may be contained in some property of directory entries. The searching is to match the queried value with the enclosed property value in the target entry item.

For the latter one, it would be much more complicated. The query can be from message content. However, message content could usually be transformed or encoded into another format, or packed into another entity. This means the query of message may be based on original message, or encoded message in the whole agent space. The original message may be typed in as a user personification according to particular user profile; while encoded message may be in the form of another syntactic pattern or semantic specification.
For simplicity, it is possible to only search for original message. To this end, it is required to design a message store structure, so that message can be stored and managed in terms of original format and business ontology as user personification. It is also required to support semantic transformation from user profile ontology to encoded message ontology. This is what ontology semantic transformation does.

6 Significance and contributions

In this section, I’ll discuss why this research work is worth time and energy. The investigation will be undertaken from research value and application value.

6.1 Research value

The main research contribution of this work is the investigation of agent services-driven approach for building open enterprise infrastructure. The proposed approach is investigated in details in terms of system modelling, analysis and design. Some key research work, for instance, user personification and domain specific ontologies, agent services-based modelling, and representation, directory, transport, transformation and mediation of agent services is studied.

An automated enterprise infrastructure supporting trading and mining is taken as instance for investigating this approach. The infrastructure prototype F-TRADE can be evolved to a test bed for multi-agent and data mining research. Without it, some of the above research cannot be undertaken effectively and efficiently. In addition, further IT-Enabled research in both theoretical and practical on capital markets can be cooperatively engaged on F-TRADE. For instance, optimization and integration of trading strategies or mining algorithms, intelligent stock selection, artificial market simulation, risk analysis, unusual pattern discovery in stock markets, mining in cross markets and the like, can be studied with supports or extensions from F-TRADE.

As the case study shows, the proposed agent services-driven approach can be used for building open complex software systems. These systems are actually open flexible enterprise systems, with capacities of interoperability, adaptability, user-friendliness, and privacy keeping.

6.2 Application value

As practical requirement engineering from capital markets, the agent services-driven F-TRADE is a web-based trading and mining support infrastructure. It supports online plug and play of trading strategies or data mining algorithms and keeps privacy. Providers can iteratively evaluate these algorithms with online connectivity to huge real stock data from global markets, and even find some optimum strategy before going to markets. Public traders and investors can benefit from interested strategies on F-TRADE by subscription. Actually, F-TRADE also supports plug and play of data sources, and system functional modules. Moreover, optimal strategies or optimal combination of input parameters for a specific trading strategy can be recommended on F-TRADE.

Further industrial applications from capital markets can be supported and expanded on F-TRADE. Some of them cannot be achieved effectively and efficiently, or even cannot be undertaken with an infrastructure like F-TRADE. Possible applications, for instance, technical analysis, fundamental analysis, risk analysis, investment risk, market replay, signal alerts, cross marketing and the like, can be added on top of F-TRADE.

With the comprehensive services support, F-TRADE can be evolved to an IT-Enabled financial platform supporting finance informationalization and decision making.

7 Evaluation
Evaluation of this work can be undertaken from the following aspects: (1) functional evaluation and comparison with other similar systems, (2) computational performance evaluation, (3) empirical evaluation from customer feedback and practical testing.

7.1 Functional evaluation

Functional evaluation of this work can be achieved from the following two aspects: (1) as a multi-agent-based system, (2) as a trading/mining supporting system. The former will be discussed from system characteristics as an agent-based system. The latter will be described from the comparison with trading system TradeStation commercially available.

7.1.1 As an agent-based system

Function requirements in Section 2.2 actually implicitly indicate that as an agent-based system, F-TRADE hold the following features.

(1) Distributed and Interoperable. It supports distributed cooperation, coordination and mediation. Related distributed components can cooperate or interoperate to achieve requested goal.

(2) Flexible. The system components are able to exhibit goal-directed behaviour, sense their environment, interact with other partners or humans, and respond in a timely fashion to changes that occur in it in order to satisfy their design objectives.

(3) Open and Dynamic. It supports active interaction between system and environment. External components can be dynamically developed or integrated by different people to achieve different objectives.

(4) Automated. The system components have their own control thread of execution, decide how and what actions to perform while situated in a particular environment at a specific time-point for achievement of a specific objective.

(5) User-friendly. Comprehensive user profiles can be supported in business-oriented format in human agent interaction; user personification-oriented interfaces can be generated in terms of particular roles.

(6) Adaptive. It has explicit capacities for supporting some potential emergent functions/applications/roles in the future, and can be expanded dynamically and easily.

(7) Privacy-keeping. This is a new requirement can be performed for particular roles, functions, resources or applications. Privacy of these stuffs can be kept on request.

These actually indicate that F-TRADE is an agent-based open and distributed enterprise system. Even though complex software systems are claimed as main research objects of multi-agent technology [35], however, an agent-based system complicated as F-TRADE is really challenging. Actually, agent-based open and distributed enterprise systems are very popular research directions in multi-agent field, especially large scale agent-based systems [70]. The research and experiment in this work shows that agent services-driven approach is powerful and flexible for building an open agent-based system. Agent services-driven approach could be a research direction and development approach for open enterprise agent-based system, in particular large scale agent systems.

These features do gestate a challenging opportunity to study new software engineering approach, and develop an infrastructure which needs to be powerful enough for supporting requests from both research and development, and from trading and mining.

7.1.2 As a trading/mining support system

Compared with all trading and mining systems available, we define some specific features for F-TRADE. They satisfy the following rules: (1) these features are determined by objectives of our investigated infrastructure, but also useful for generic development, (2) most of them can benefit both trading and
mining. Some main functions which can be available for supporting trading and mining in FTRADE include the follows.

(1) Supporting plug in and online registration of data sources, system modules, and algorithms etc. This means that users, developers and providers can plug in data sources, algorithms and system modules into FTRADE, and FTRADE will register them automatically. The pre-condition is the plugged components of algorithms and system modules must implement the API provided by FTRADE.

(2) Providing data gateway for supporting data linkage and cross markets. Link and data source information can be set for remote data sources, data crossing markets can be accessed by algorithm or data services.

(3) Supporting user profiles-oriented and problem domain-oriented interaction. The target for this is to support user-friendly and business-oriented user personification, rather than technology-centered or symbol-centered user interface interaction. User profiles and ontologies for financial trading will be defined and used as interactive language.

(4) Supporting optimization and integration of trading rules. Optimal value candidates of a parameter or optimal combinatorial value candidates of parameters can be recommended according to user portfolio. Stock watchlist can be recommended based on integrated benchmark of multiple trading rules. We are investigating whether it is feasible for this feature to be used for data mining algorithms.

(5) Supporting online system customization and reconstruction. FTRADE can be customized according to user profiles and permission. System components and organizational relationship can be re-organized into another organization.

(6) Supporting comprehensive add-on applications from capital markets. Comprehensive research and development, for instance, technical analysis, fundamental analysis, e-training and learning, investment analysis, unusual trading patterns discovery, can be supported by and integrated into FTRADE.

More specifically speaking, as discussed before, the infrastructure under investigation is mainly aiming at as an IT-Enabled financial research and trading support platform. So, here I take this of its roles as observation point, and mainly discuss the functional features as a trading system through comparison. I still take the commercial trading system TradeStation as comparable partner of FTRADE. The first reason is TradeStation is the most similar system to mine. The second reason is it has been widely used in the world. Since its introduction in 1991, TradeStation has got choice for over 30,000 traders in over 25 countries worldwide. TradeStation continues to win Stocks & Commodities Magazine’s Readers’ Choice for Best Trading Software for the last six years.

I compare these two systems from the following main functions: (1) trading supports, (2) data supports, and (3) system supports. This is shown in table 1.

Table 1 Comparison between TradeStation and F-Trade

<table>
<thead>
<tr>
<th>Function Items</th>
<th>TradeStation</th>
<th>F-TRADE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Back-testing</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Indicators</td>
<td>Yes</td>
<td>Yes, but not enough</td>
</tr>
<tr>
<td>Basic charting</td>
<td>Yes</td>
<td>Yes, but not strong</td>
</tr>
<tr>
<td>Reporting</td>
<td>Yes</td>
<td>Yes, but not strong</td>
</tr>
<tr>
<td>Draw tools</td>
<td>Yes</td>
<td>N/A, Can support</td>
</tr>
<tr>
<td>Signal alert</td>
<td>Yes</td>
<td>N/A, Can support</td>
</tr>
<tr>
<td>Automated execution</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Feature</td>
<td>F-.TRADE</td>
<td>F-TRADE with potential support</td>
</tr>
<tr>
<td>------------------------------</td>
<td>----------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>Market replay</td>
<td>N/A</td>
<td>N/A, Can support</td>
</tr>
<tr>
<td>Stock recommender</td>
<td>N/A</td>
<td>N/A, Can support</td>
</tr>
<tr>
<td>E-training &amp; learning</td>
<td>N/A</td>
<td>N/A, Can support</td>
</tr>
<tr>
<td>Technical analysis</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Fundamental Analysis</td>
<td>No</td>
<td>N/A, Can support</td>
</tr>
</tbody>
</table>

**Data supports**

<table>
<thead>
<tr>
<th>Data formats</th>
<th>Some commercial stock data</th>
<th>JDBC, ODBC, FAV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intraday</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Inter-day</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Real time</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Downloadable</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Cross markets</td>
<td>N/A</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**System supports**

| Web-based                     | Yes                           | Yes             |
| Downloadable pack             | Yes                           | N/A             |
| Formula disclosed             | No                            | Can support     |
| Rule customizing              | Yes, using EasyLanguage      | N/A             |
| Rule optimization             | Yes                           | Yes             |
| Rule integration              | No                            | Yes             |
| Inner language                | Yes, EasyLanguage            | N/A             |
| Data plug in                  | No                            | Yes             |
| Module plug in                | No                            | Yes             |
| Algorithm plug in             | No                            | Yes             |
| Interface creating            | Yes                           | Yes, automated |
| Data gateway                  | No                            | Yes, data link and access |
| System reconstructing         | No                            | Yes             |
| Multi-level outputs           | N/A                           | Yes             |
| Visual output                 | Yes                           | Yes, not strong |
| System logging                | Yes                           | Yes, Algorithm, system |
| Privacy keeping               | N/A                           | Yes             |

Even though F-TRADE is not strong enough as a commercial platform like TradeStation, but as a prototype, it has shown some particular and promising functions as we can find from the above table. These functions can be taken as foundation supporting further potential applications from capital markets. Some ongoing work, for instance, investment decision support system investigated by another PhD student of the Data Mining Program at CMCRC, trading education system activated by a broker from TriCom[71], intelligent stock selection using technical analysis signals by another PhD student using data mining technology, is conducted on top of F-TRADE.
7.2 Empirical evaluation

Empirical research of the proposed infrastructure will be undertaken from some of the practical testing of the proposed functionalities in industry or other real-world environment through the wide area network. This practical testing is not only for evaluating whether the key functions claimed work or not in the real world, but also estimates the computational performance for settling down these functions in the real environment.

The computational performance of F-TRADE will be calculated, if possible, from the following aspects: (1) runtime cost spent for executing each of system functions, (2) I/O cost for executing each of these functions.

The scenarios of the testing environment are shown in Figure 2. F-TRADE is currently located on Broadway Street at Faculty of IT, University of Technology Sydney. Remote stock data is situated in data warehouse of AC3 [72] at Australian Technology Park at Redfern Railway Station. Our testing node was at office of CMCRC, CBD of Sydney.

![Figure 2. Testing environment of F-TRADE](image)

The running environment is as follows. F-TRADE is running on a DELL workstation with the following specifications: 2 x 2.40GHz CPU (Intel(R) Xeon(TM)) with 2GB ECC DDR SDRAM and 147 GB SCSI drive (FUJITSU MAP3147NP). On this server, Windows 2000 professional is installed on top of Linux Red Hat 7 with virtual machine support of VMWare. IBM DB2 7.2 for Windows is used for system and local stock databases. Components for dealing with channel link and communication, data extraction and transferring between remote data sources and F-TRADE is conducted by C components in Linux. Part of these C components is located on the remote data sources, some others are situated in the Linux box of F-TRADE.

Some key functions proposed in this infrastructure have been tested in the practical environment. I’ll further summarize the empirical testing results in terms of Workable or Not and computational Spending. Computational Spending is conducted from Runtime in seconds and I/O cost.

8 Conclusions
This work is based on a practical and feasible exploration in the following scenarios: (1) problems are extracted from the real and complicated requirements in capital markets; (2) data is from the huge real stock data remotely situated at AC3. In this work, the following problems have been investigated: (1) setting up an automated enterprise infrastructure integrating both stock trading and data mining; (2) exploring a new agent-based design paradigm called agent services-driven approach for building open agent-based systems; (3) building an agent services-driven infrastructure supporting trading and mining. The following is the summarized remarks of these three aspects.

8.1 An automated enterprise infrastructure integrating both stock trading and data mining

In the CMCRC project, it is required to build an infrastructure so that research and applications in capital markets can be supported on it. As the discussions in the above sections, this infrastructure should be (1) automated and enterprise-oriented, (2) a support workspace both for stock trading and data mining. The main functions of this infrastructure include supports for user personification and financial domain-oriented interaction, online plug and play of algorithms or system components, connectivity with huge real stock data, iterative backtesting, recommendation and integration of trading strategies or mining algorithms, online system customization and re-construction, and add-ons of applications from capital markets.

The structure of the infrastructure is actually distributed and multiple-level-based. Multiple heterogeneous data sources, some are local some are dynamically remote, are connected and integrated on demand. There are multiple levels from one-stop interface to underlying data resources. They are user personification-oriented human-system interaction, industrial applications-oriented enterprise interaction, customized and categorized function centers, web server for web-based driven, services-driven engines for supporting system/algorithm/data services and application mediation, resource and data gateway to data sources/algorithm, and system and data resources, from top down.

There are five function Centers in F-TRADE. They are Administration Center, Algorithms Center, Control Center, Services Center, and User Center. These function centers are one-stop interfaces for user-system interaction dealing with different categorical requirements.

The system prototype and empirical evaluation have shown that it is necessary and feasible for us to build an infrastructure integrating both stock trading and data mining. It is actually an automated enterprise system in terms of the comprehensive requirements and existent environment in the real capital markets. This kind of system can benefit different system users like brokers, retailers, financial researchers, data miners, multi-agent researchers. It is also a test platform for enabling finance through IT technology.

8.2 Agent services-driven approach as design paradigm for building open agent-based systems

Agent-oriented methodology is suitable for system analysis and design of the above proposed infrastructure. In order to build the proposed automated enterprise infrastructure, agent services-driven approach has been studied as a complete design paradigm for building an open agent-based system in terms of the real environment and requirements. The investigation has been undertaken from modeling, analysis and design of an agent services-based system.

As a complete design approach, the following problems have been investigated in details: (1) finance domain specific ontologies and semantic relationships, (2) the naming and organization of the agents and services, (3) representation of agent services, (4) modeling of agent services-based problem, (5) mediation of requests and response in the agent space, (6) location and transportation of agent services, (7) transformation of agent ontology and services in the agent ontology space.

The investigation of this approach is worthwhile. It could be a new approach for building practical open and enterprise agent-based system. This type of systems could be a very large agent system. Research on this type of systems is very challenging and popular in the agent research field.
8.3 Agent services-driven enterprise infrastructure supporting trading and mining

According to the above investigation, Java agent services has been studied as developing tool for building an agent services-driven enterprise infrastructure. An infrastructure prototype -- F-TRADE (Financial Trading Rules Automated Development and Evaluation) -- has been built in terms of the above requirements and observations.

Functional comparison and empirical research have shown it is feasible for agent services-driven approach to build an open agent system. It also has shown that the agent services-driven infrastructure can be powerful enough for supporting both trading and mining in capital markets.

F-TRADE is actually evolving to be an online test platform for research, application and integration of multi-agent technology and data mining in stock markets, and linking finance and IT to fulfil IT-Enabled Finance in E-Intelligence Group of UTS and Data Mining Program in CMCRC.

This exploration has further shown us that the agent services-driven approach is powerful for building open enterprise complex software systems. This could be a solution for the challenging issue in building large scale open agent system.

9 Future work

My remaining work will be on the following in terms of the current work progress and overall plan.

(1) Refinement of agent services-driven approach

This includes work of system modelling using agent services, location and transportation of agent services, transformation of agent ontology and services in the agent ontology space in multiple data sources.

Another investigation will be done is comparison between web services and agent services. I’ll investigate whether agent services-driven approach can benefit from web service technology.

Model integration using agent services will be investigated for integrating multiple trading strategies or data mining models.

(2) Thesis and publication preparation

This mainly includes writing the thesis, and generating some publications if possible.

10 Timetables

This report sums the essential contents and profits of my PhD research work. It presents what has done in the past one year and points out my proposals to the potential research topics. The following approximate timetable provides an entire schedule for the research as shown in table 2.

<table>
<thead>
<tr>
<th>Time</th>
<th>Research Work</th>
</tr>
</thead>
<tbody>
<tr>
<td>March, 2003 ~ August, 2003</td>
<td>- Project requirement analysis</td>
</tr>
<tr>
<td></td>
<td>- Literature review</td>
</tr>
<tr>
<td></td>
<td>- System review</td>
</tr>
</tbody>
</table>
The section list the content of the thesis, and publications up to now.

11 Thesis content

1 Introduction
1.1 What’s the problem?
1.1.1 How does the problem emerge?
1.1.2 What specific problems from financial markets
1.1.3 What’s my research problem?
1.2 Integrated infrastructure supporting trading and mining
1.2.1 Can trading and mining be supported in one system?
1.2.2 What a system supporting both trading and mining is expected
1.3 What are main research objectives?
14. The organization of the thesis

2 Related work
2.1 Similar systems
2.1.1 System classification
2.1.2 Similar systems
2.2 Research methods
2.2.1 Objects, components, services and agents
2.2.2 Object-oriented, component-based, service-oriented, agent-oriented

3 Agent services-driven approach
3.1 Research methodology in my work
3.2 Agent-oriented methodology
3.3 Agent services-driven approach
3.3.1 The agent services-driven approach
3.3.2 Why agent services-driven approach

4 Agent services-oriented requirement analyses
4.1 Requirement engineering
4.1.1 Enterprise application integration
4.1.2 Plug and play
4.1.2 Enterprise portal
4.2 System functions
4.2.1 Trading/mining supports
4.2.2 Data supports
4.2.3 System supports
4.3 Conceptual model
4.4 Goal-oriented modeling

5 Ontology and representation of agent services
5.1 User personification
5.2 Agent ontology
5.3 Semantic relationships
5.4 Ontology transformation
5.5 Agent services naming
5.6 Representation of agent services

6 Agent services-oriented system analyses
6.1 Role models
6.2 Interaction models
6.3 Environment model
6.4 Organizational Rules
6.5 Organizational Structure

7 Agent services-oriented system design
7.1 Agent service model
7.2 Agent organizational architecture
7.3 Agent design patterns
7.4 Agent service-oriented architecture

8 Organization and management of agent services
8.1 Directory of agent services
8.2 Communication of agent services
8.3 Transport of agent services
8.4 Mediation of agent services
8.5 Discovery of agent services

9 A prototype system: F-TRADE
9.1 Development environment
9.2 Functional screenshots
9.3 Programming interfaces
9.4 Knowledge portal

10 Agent-based trading support system
10.1 Trading support system
10.2 Agent-based trading system

11 Agent-based mining support system
11.1 Mining support system
11.2 Agent-based mining system

12 Evaluation and testing
12.1 Functional evaluation
12.1.1 As an agent-based system
12.1.2 As a trading/mining support system
12.2 Empirical evaluation

13 Conclusions
13.1 Research value
13.1.1 Agent services-driven approach as design paradigm for building open agent-based systems
13.2 Application value
13.2.1 An automated enterprise infrastructure integrating both stock trading and data mining
13.2.2 Agent services-driven enterprise infrastructure supporting trading and mining
13.3 Future work

Publications
References

12 List of related publications


References

[1] datamining.it.uts.edu.au
[3] datamining.it.uts.edu.au:8080/tsap
[61] SC00067F. FIPA Agent Message Transport Service Specification. 2002/12/03
[71] www.tricom.com.au