The OSOAD Methodology for Open Complex Agent Systems

Longbing CAO, Chengqi ZHANG, Ruwei DAI

Abstract- **Open complex agent system (OCAS) are middle-size or large-scale open agent organization. To engineer OCAS, agent-centric organization-oriented analysis, design and implementation, namely organization-oriented methodology (OOM), has emerged as highly promising direction. A number of OOM-related approaches have been proposed; while there are some intrinsic issues hidden in them. For instance, some fundamental system attributes, such as system dynamics, are not covered by almost all of existing approaches. In this paper, we summarize our investigation of existing approaches, and report a new OOM approach called *OSOAD*. The OSOAD approach consists of organizational abstraction (OA), organization-oriented analysis (OOA), agent service-oriented design (ASOD), and Java agent service-based implementation. OSOAD provides complete and deployable mechanisms for all software engineering phases. Specifically, we notice the transition supports from OA to OOA and ASOD. This approach has been built and deployed with practical development of agent service-based financial trading and mining applications.**

***Index Terms*—Open complex agent system, organization oriented analysis, agent service oriented design**

1. INTRODUCTION

Open complex agent systems (OCAS) [3], for instance, an online agent-based financial trading systems, are middle-size or large-scale open agent systems [14]. OCAS are critical because they are used to deal with those complex problems that cannot be handled by traditional technologies or simple agent systems.

In the research on OCAS, engineering OCAS has become a fundamental task in building OCAS; while it is also challenging due to the intrinsic system complexities in an OCAS [3]. As a highly promising direction to engineer OCAS, a number of agent-oriented software engineering (AOSE) researchers have referred organizational theory [7] and artificial social systems [8] to seek original and effective support. This trend builds the agent-centric organization-oriented analysis, design and implementation, namely *organization-oriented methodology* (OOM) [3]. Some typical OOM-related approaches include Formal TROPOS [15], GAIA [23], MASE [24], MESSAGE [2], OMNI [11], ROADMAP [17], SODA [19], etc.

However, there exist many issues in existing OOM-related methodologies and approaches [10, 3]. For instance, there are big disagreement among varied approaches in aspects such as modeling concepts and techniques. More importantly, some important system attributes, for instance, system dynamics, are not covered by almost all of existing approaches.

In more than six years work developing an agent-based macro-economy decision support system [4, 5], a financial trading and mining infrastructure F-Trade [6, 13] for the research and industrial development at Data Mining Program of Capital Markets Cooperative Research Center (CMCRC), and projects from Credit Swiss First Boston (CSFB), we targeted and built a practical organization-oriented AOSE approach and its model-building blocks [3]. Unfortunately, space limitations preclude a detailed description of this approach, but here we summarize our major ideas in building the approach, referred to as *Organization and Service Oriented Analysis and Design* (OSOAD) [3]. It consists of full mechanisms for *organizational abstraction* (OA), *organization-oriented analysis* (OOA), *agent service-oriented design* (ASOD), and *Java agent service-based implementation* (JASBI).

The main contributions of the OSOAD include the following aspects. (i) It proposes a systematic abstraction framework named ORGANISED which explicitly captures almost all main attributes in an OCAS; while some of the ORGANISED elements such as system dynamics have not been covered by other approaches. (ii) The OSOAD, to the best of our knowledge, is the first work in the world which integrates organization-oriented modeling and service-oriented computing. The dialogue between them leads to bilateral benefits for the abstraction and modeling based on organization-oriented modeling, and the system design of large-scale distributed software organization utilizing service-oriented computing. (iii) Compared with other OOM-related approaches, our OSOAD embeds complete and deployable integration methodology and model-building blocks for all software engineering phases, supporting smooth transition from OA, OOA to ASOD, and undertaking practical implementation based on Java agent service.

The rest of the paper is organized as follows. Section 2 discusses issues in existing OOM-related approaches. The OSOAD approach is outlined in Section 3. Section 4 presents organization-oriented analysis. In Section 5, agent service-oriented design is introduced. Section 6 discusses the transition from OOA to ASOD. Section 7 introduces agent service implementation strategies and the case study system. Section 8 presents related work and discussion. Finally, Section 9 concludes this research.

2. ISSUES IN OOM-RELATED APPROACHES FOR ENGINEERING OCAS

Complexities of OCAS are embodied via characteristics such as a great many of system constituents, openness, distribution, interaction, hierarchy, integration, socialization, uncertainty, and human involvement [3]. For instance, in an OCAS, goals would be multiform and/or hierarchical; its environment is often open, networked, heterogeneous or uncertain; the interactions, agent activities and behaviors take place temporally, spatially, or

spatial-temporally; the constraints and the rules on social interactions may present at multiple dimensions; in addition, the system structure would be rather dynamic and complicated.

In the investigation of existing OOM-related AOSE approaches [3], we find out some critical issues from the OOM perspective. The following lists some of them.

1) Most existing approaches are not developed in terms of OOM, they are created based on either traditional agent theory or non-AOSE requirements.

2) The organizational abstraction is not yet complete in most of them. Some important organizational attributes such as environment and organizational dynamics haven't been covered in most approaches.

3) Every phase of OOM hasn't been investigated clearly enough. For instance, some phases are not supported, architectural and detailed designs are merged.

4) None of existing approaches provides transition support among the varied phases in OOM. For instance, the transition from system analysis to design is not studied, implementation techniques are not touched in most approaches.

Most approaches develop either graphical or formal specifications for modeling. Nevertheless, formal specifications and integrative modeling embedding both visual and formal specifications are essential for complete and precise analysis.

3. THE OSOAD METHODOLOGY

With the above issues in mind, we built a new OOM approach called *OSOAD*. A key component in the OSOAD approach is the *ORGANISED* framework, which abstracts all main system attributes in an OCAS. This section first introduces the ORGANISED, and then presents a brief overview of the OSOAD approach.

3.1 The ORGANISED Framework

Taking the organization-oriented philosophy for engineering agent organizations, an OCAS is abstracted and modeled as an artificial organization in terms of human organizations and organizational theory [6]. Furthermore, the modeling of OCAS can also benefit from other multiple disciplines such as system sciences and the science of complexity [20] for deep understanding the OCAS, for instance the system complexities and dynamics [3]. As a consequence, the OOM captures all major intrinsic attributes in an agent organization.

Following the above thinking, and the investigation of existing OOM-related AOSE approaches [3], we find out that there are big disagreement among varied approaches in aspects such as major system attributes, and modeling concepts and techniques. More importantly, some important system attributes, for instance, system dynamics, are not covered by almost all of existing approaches. Therefore, we propose a new framework, referred to as the ORGANISED framework, which targets a unified and relatively complete organization-oriented view of OCAS.

The ORGANISED framework consists of the following fundamental system attributes: Organization, Rules, Goals, Actors, Norms, Interactions, Structures, Environment and system Dynamics (the capitalized letters form the name ORGANISED). This framework synthesizes some key but generic concepts such as actor available from existing approaches, in particular the meanings and ranges of some members have been expanded. For instance, in our framework, an actor may take form as an agent, service, workspace or human. In addition, environment and dynamics, two essential attributes in OCAS, have been highlighted in this framework. Table 1 lists the definitions of these attributes. Figure 1 further shows the metamodel of the ORGANISED framework.

Table 1. The ORGANISED framework members

Member Name	Description
Organization	An OCAS is an artificial organization; the organization-oriented abstraction and analysis explicitly adopt the organizational metaphor
Rule	Organizational rules are temporally and/or spatially distributed over an organization managing its actors, activities and evolution; this work focuses on two types of rules i.e. structural and problem-solving rules because they are fundamental for agent-based system evolution and problem solving
Goal	A goal is certain overall common motivation and objective of an organization, or individual target of a sub-organization or an actor; from the perspective of designing, goal consists of functional and nonfunctional objectives of an OCAS
Actor	Actors may be active or passive stakeholders or abstract concepts playing different roles at varied tiers of an organization, including human, workspace, autonomous, service and resource actors
Norm	Norms may be presented as social patterns governing perceptual, denotative, evaluative, cognitive or behavioral aspects
Interaction	An interaction is a social activity at certain granularity in an organization in which certain organizational relationship acts on specific actors following some organizational rules, for instance, inter-role (or inter-actor) negotiation, mediation, teamwork, coalition, resource access, and conflict resolution, etc.
Structure	Organizational structure is an collectively emergent architecture-oriented pattern from the interaction among system members at corresponding system level following certain rules
Environment	Environment is a relative object that may comprise actors, principles, processes and forces [16] inside or outside an agent system or its subsystem; the features of environment, i.e. accessibility, determinism, uncertainty, diversity, controllability, volatility, continuity, locality, temporality or spatiality determine whether a system is open, semi-open, semi-closed or closed [3]
Dynamics	Organizational dynamics is a collective emergence of all stakeholders interacting in light of the rules and goals of an organization, which leads to the overall behavioral patterns, swarm intelligence

	emergence and problem solving of the organization; from system science perspective, an organization may take the form as a static, a discrete-event dynamic, or a continuous-time dynamic system following patterns such as self-organizing, center-controlled, or stochastic bodies, etc.

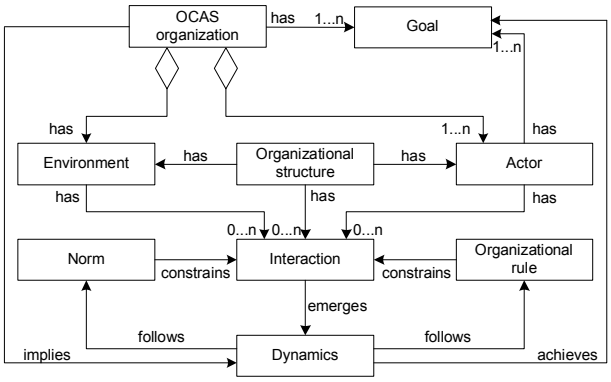

Fig. 1. The ORGANISED metamodel

3.2 The OSOAD Methodology

The OSOAD approach consists of software engineering mechanisms for all software engineering phases including early requirement analysis, OOA for organizational abstraction and late analysis, ASOD for system architectural and detailed design, and JASBI for implementation. We pack all above and call it as *Organization and Service Oriented Analysis and Design* (i.e. OSOAD approach).

- The early analysis discovers integrative requirements including goal-oriented requirements and business-oriented requirements. Both of them cover functional and nonfunctional requirements. These requirements extracted are used to abstract the domain problem and problem-solving system. In addition, the ORGANISED framework is recommended to capture all main organizational attributes such as goals, rules, actors, norms, interaction, structure, environment and dynamics in an OCAS. The output of this phase is an ORGANISED model.
- The late analysis (i.e. the main task of the OOA) builds individual models for all elements in the ORGANISED model.
- The architectural design is achieved through building (i) agent service abstract model, (ii) agent service design pattern, (iii) agent service management, (iv) agent service communication, (v) interaction pattern, (vi) information integration pattern, (vii) system architectural frameworks, and (viii) integration level and strategy.
- In the detailed design, models built in the architectural design are instantiated into (i) agent service ontology, (ii) agent service specification, (iii) agent service endpoint interface, (iv) the transport, directory, mediation, management and communication of agent service, and (v) data structure model, etc.
- The agent-based problem-solving system is implemented in Java agent service. The following components are instantiated: (i) interface agent, (ii) configure agent, (iii) register agent, (iv) data gateway agent, (v) ontology service, (vi) directory service, (vii) transport service, (viii) directory service, (ix) discovery service, etc.

To advice the above lifecycle, according to our empirical practice in developing the F-Trade, the following briefly lists hybrid policy and philosophy in capturing the ORGANISED members, and analysis and design of an OCAS.

- Reductionism for top-down decomposition – the reductionism philosophy is taken in the decomposition; it is recommended from empiricism to capture high-level attributes such as goals and structures first, and then go deep to analyze rules, interaction, dynamics and actors hidden in the system.
- Holism for bottom-up aggregation – it is recommended to go up from the decomposition to get a unified or overall view of the organization taking a holistic policy; this is helpful for understanding subsystem-level and system-level characteristics by aggregating components into subsystems, architectural frameworks, or design patterns.
- Systematology [18] for iterative refinement and integration – the process for decomposing and aggregating an OCAS would be iterative, it is recommended to undertake progressive refinement of the decomposition and aggregation, and finally form the overall integrated ORGANISED framework, and analysis and design models via taking the philosophy of systematology.

In practice, the above software engineering process is also iterative rather than linear as in a waterfall model.

4. ORGANIZATION-ORIENTED ANALYSIS

4.1. Analysis Process

In the OOA, the following aspects are specified. A simplified OOA activity list is shown in Table 2.

- Organizational abstraction and decomposition in terms of the ORGANISED framework;
- Regarding functional requirements, building model-building blocks for all elements in the ORGANISED model;
- For nonfunctional requirements, developing system supports for flexibility, autonomy, proactive ability and other qualities the system must hold;
- Presenting the above two types of requirements in diagrammatic notations and/or formal specifications;

- Analyzing evolutionary activities and process of the target organization by developing scenario description and scenario-based analysis;
- In order to capture the state transition of stakeholders, listing state chain sequence of each stakeholder to grasp the state transition in the organization.

Table 2. key activities and questions in the OOA

OOA main activities	key questions
Gather information	Do we have all of the information, e.g. goal, belief, intention, insight, desire and environment we need to define what the system must do?
Define system requirements	What functional (stakeholders, goals, rules, policies, constraints, etc) and nonfunctional (global qualities of the system) details do we need the system to do?
Prioritize requirements	What are the most important goals and interaction activities the system must do?
Prototype for feasibility and discovery	Can the organization-oriented model-building technology proposed deal with what we think we need to do in the system? Have we built some prototypes to ensure that the users fully understand the potential of what the technology can do?
Generate and evaluate alternatives	What's the best way for organization-oriented analysis to develop the agent-based system?
Review recommendations	Should we continue and design and implement the system we propose?

4.2. Modeling ORGANISED Members

We have built model-building blocks for all ORGANISED members. They are goal model, actor and role models, structural and problem-solving organizational rules, modeling norms, interaction model, modeling organizational structure, environment model, and organizational dynamics analysis model. As an instance, the following briefly exemplifies how to analyze organizational dynamics.

In [3], we demonstrated some techniques for analyzing organizational dynamics. Here we take the $POMDP_{AEI}$ model [22] as an example. In the F-Trade, the goal of agent *AlgoPluginAgent* is to register an algorithm into the system. Corresponding environment states and actions of this agent are listed in Table 3 and 4, respectively. The state-action chain of this agent interacting with its environment can be further modeled as Figure 2. In addition, this state-action chain can be formally specified. For instance, the state transferring from s_1 to s_2 under condition a_1 is represented as follows ($t_1 < t_2$).

- $\exists$ apr: AlgoPluginRequest (apr.depender = **PLUGINPERSON** $\wedge$ apr.dependee = PluginInterface $\wedge$ Fulfilled() $\rightarrow$ $[\Diamond_{\leq t2}$ CheckAlgorithmValidity$]_{0 \leq t1}$ AcceptPluginRequest.Fulfilled()

Table 3. *AlgoPluginAgent* environment state list

State	Description
s_1	registration request by PluginInterface (A) agent
s_2	accessible algo model(B) and ontology (C) bases
s_3	none record of the requested algorithm in base B
s_4	algorithm ontologies typed by PluginInterface
s_5	algorithm ontology base (C) is accessible
s_6	configuration table (C) of ontologies exists
s_7	data source management base (D) is accessible
s_8	accessible local(E) and remote(F) data sources
s_9	specific sources configured by PluginInterface
s_{10}	none record in the XML configuration files (G)
s_{11}	open connections to all information bases BCD
s_{12}	register result available to PluginInterface BCD
s_{13}	connection closed to all information bases

Table 4. *AlgoConfigureAgent* action list

Action	Description
a_1	receive register request from PluginInterface
a_6	register and configure ontologies into base C with help from OntologyService (H)

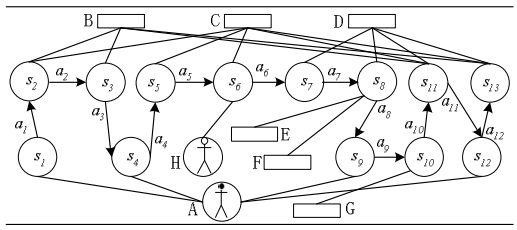

Fig.2. POMDP state chain for *AlgoConfigureAgent*

5. AGENT SERVICE ORIENTED DESIGN

The *Agent Service-Oriented Design* is composed of two levels of system design in defining the system design mechanisms – namely *agent service-oriented architectural design* and *agent service-oriented detailed design*. Notice that this process is iterative as recommended to follow the hybrid philosophy.

5.1. Agent Service-Oriented Architectural Design

In agent service-oriented architectural design, a set of models are built, specifically the following highlighted.

- Agent Service Abstract Model: This defines the types of agent and service architecture, and properties and attributes of agent services.
- Agent Service Design Pattern: It abstracts some high-level agent architectural frameworks, and structural and functional service patterns.
- Agent Service Integration Strategies: Two strategies for implementing agent service-oriented computing are studied; they are "multiagent + Web services" and "multiagent + service-oriented computing".
- Agent Service Integration Architectures: General integration levels and techniques, and system architectural frameworks are specified for agent service-oriented architectures.
- Agent Service Management and Communications: The main issues related to management and communications of agent services include naming,

directory, communication, transport, mediation, and discovery of agent services in integrated enterprise applications.

For instance, in order to integration some legacy applications, we develop agent services such as proxy, wrapper and adapter for atomic transactions, gateway for data access, and mediator and matchmaker for agent service management. In an agent service-enabled hub and spoke architecture [3], a centrally located server hosts the integration logic that controls the orchestration and brokering of all inter-application communication. Application 1 is integrated after wrapped as a service; application 2 interacts with a wrapper server via an agent adapter; both of them interact with the server through business services. In addition, application 3 and 4 are combined with the server via an agent adapter and a gateway agent respectively.

5.2. Agent Service-Oriented Detailed Design

Agent service-oriented detailed design focuses on developing the internal structure of each agent and service, and mechanism for their interaction to achieve its goals. In this phase, a number of models are developed. We specifically highlight the following modeling tasks by tracing the models in architectural design.

- Agent Service Ontology: This aims to build generic glossary and semantic relationships of ontological items in agent service-oriented computing for a specific problem domain.
- Representation of Agent Services: This defines a specification for presenting agent services.
- Agent Service Endpoint Interfaces: It outlines some principles for designing generic agent service interfaces.
- Agent Service Organization and Management: The organization and management of agent services involves mechanisms for directory, communication, transport, mediation and discovery of agent services.
- Agent Service-Oriented Design Strategies: Here some strategic issues in designing agent service-oriented system are specified.

For instance, in order to design enterprise-wide generic and consistent interfaces, we undertake the endpoint interface design. This first establish generic and consistent enterprise-wide naming conventions and interface design standards; an integration layer provides standard interfaces to disparate applications. Then we can encapsulate business logics using agent services. However, there are multiple strategies for us to implement this. For instance, one way is that an agent service provides a generic operation (service interface) that represents multiple agent service methods. As illustrated in Figure 3, in which the operation Implementing Aglorithm (ImplementAlgo) is performed via methods Implementing Algorithm API (ImplemtAlgoAPI) and Coding Algorithm Logics (CodeAlgoLogic).

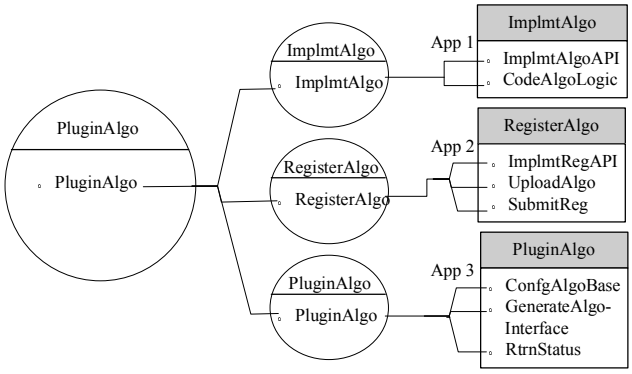

Fig. 3. An agent service with multiple operations via combining multiple agents' methods

In addition, besides the issues addressed in the above discussion about architectural and detailed designs, following aspects will further be investigated in detail in order to build agent services-oriented design approach: (i) agent service ontology and representation, (ii) naming and registry of agent services, (iii) agent service discovery.

6. INTERACTION BETWEEN OOA AND ASOD

After building the system for OOA and ASOD, we need to consider how to make a smooth transition from OOA to ASOD. The first is about the dialogue between agents and services; we will not discuss it for space limitation, detailed information is available from [3]. The second is about the interaction between organization and service. The dialogue between organization and agent service is based on the following observations and linkage.

- Software engineering theorists appeal to the concept of human organization to view and understand an agent system. In this sense, an agent system is an artificial organization. An artificial organization is an emergent presentation of interaction among all system members and its environment.
- However, agent service is a kind of computational concept that is used to describe computational entities in a problem-solving system. It embodies the characteristics of system members; it also encompasses computational behaviors and problem-solving capabilities in terms of both intra- and inter-agent services.

Furthermore, the following technical mechanisms support the interaction and transition between organization-oriented analysis and agent service-oriented design.

- The findings of OOA serve as the base of ASOD. ASOD cares about how to embody and implement those system elements, behaviors and objectives captured in OOA.
- All findings in OOA will be further embodied and instantiated into computational entities and structures supported and implemented in ASOD. The elements in

OOA could be explicitly or implicitly mapped to elements in ASOD in some manner.

For instance, the following Table 5 lists the mapping between some elements in OOA and corresponding items in ASOD discussed.

Table 5. Mapping between OOA and ASOD

Organization-oriented analysis	Agent service-oriented design
Actors	Agents, services, sources, agent service architecture
Environment	Agents, services, sources
Organizational rules	Structural and semantic relationships in agent service ontologies, strategies and behavior rules of agent services
Organizational structure	Agent service architecture, system architecture
Organizational dynamics	management and communication of agent services

7. IMPLEMENTATION AND EVALUATION

The implementation of an OCAS is recommended to be in Java agent service. Java agent service can be undertaken in terms of two kinds of strategies and techniques.

7.1 Agent Service Integration Strategies

7.1.1. Multiagent + Web services

This approach builds a dialogue between multiagent system and XML-driven Web service (from hereon referred to as *Web service*) [12]. The integration focuses on building enterprise integration infrastructure and integrated applications, agent-based dynamic service integration, agent-based automatic negotiation in Web services, conversations with Web services, agents for Web service composition, etc. We call this approach as "Web services-driven agent systems". In this method, we advocate the usage of the second-generation Web services.

The basic work for the Web service-based infrastructure includes:

- Provide a service description that, at minimum, consists of a WSDL document;
- Be capable of transporting XML documents using SOAP over HTTP.

On the other hand, multiagent technology would play significant roles in aspects beyond the above basic infrastructure linkage. For instance, the following lists some main functionality that multiagent can serve.

- Agentized components: applications and application components in integration enterprise can be agentized on demand for establishing some specific functional components. For instance, agents for human-computer interaction, resource access dispatching, management of Web services, and enterprise business logic, etc. The problem here may include how to link and administer these agents in the Web services-centric environment.
- Management, dispatching, conversation, negotiation, mediation and discovery of Web services: for these issues, multiagents can play great roles with flexible, intelligent, automated and (pro-) active abilities. For instance, a gateway agent located at a remote data source listens to requests and extracts data from the source after receiving data request messages, it generates and dispatches an agent to deliver the extracted data to the requestor after completion. This can reduce the network payload and enhance the flexibility of remote data access.
- Agent-based services: services for middlewaring and business logics such as adapter, connector, matchmaker, mediator, broker, gateway and negotiator can be customized as agents. For instance, an agent-based coordinator fulfilling partial global planning has the capability to generate short-term plans to satisfy themselves, it may further alter local plans in order to better coordinate its own activities. Agent-based services can enhance the automated and flexible decision-making of these services.

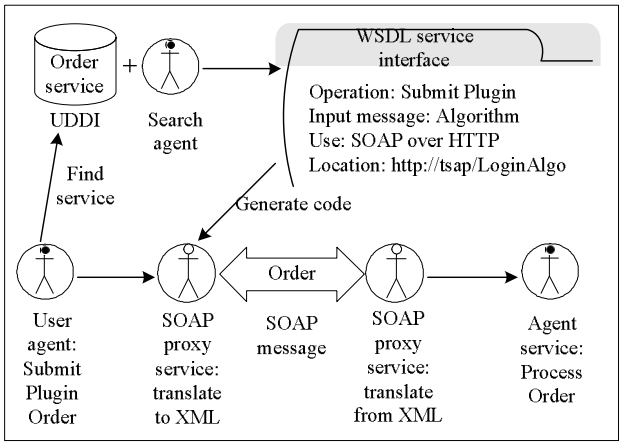

Fig. 4. Web services-driven agent systems for EAI

Figure 4 describes the architecture of a Web services-driven agent system. In this system, client defines and submits algorithm Plugin Order via user agent. User agent finds Order Service through UDDI and Search agent. A SOAP proxy service is generated by using WSDL. This proxy service is taken as a bridge to talk to the Process Order agent service via SOAP message and a proxy service translating the message.

7.1.2 Multiagent + service oriented computing

This strategy is based on the combination of Multiagent and service-oriented computing. Service-oriented computing is not necessary to be Web service-driven or -centric. In this approach, multiagents act as the main building blocks of a system. The system following this approach is called as "Multiagent-driven service system". The fundamental theory of the multiagent-driven service systems is as follows.

- The system analysis is performed in terms of agent-oriented early and late requirements analyses, in particular the fore-detailed organization oriented analysis;
- The system architectural design is undertaken in light of agent service-oriented design, especially the fore-mentioned agent service-oriented architecture and application integration;
- The system detailed design is achieved via agents and services, and agent service-oriented interactions;
- Agents consist of the main building blocks in the system; while services play significant roles such as services of agents, services of services, inter-application communications and integration, etc.;

By contrast, the content of services here is a little different from the services in Web services. This can be embodied in terms of two aspects: (i) the conceptual scope is broader than that of Web services, for instance, services could be of agents and services; (ii) agent service is a unified concept and a computational entity, agents encompass explicit service attributes and features, services are agentized through supporting autonomy, flexibility and proactive capability.

The above discussion actually consists of the main theory of "Agent Service Oriented Computing" we advocate and detail in our work [3].

7.2. Case Study and Evaluation

We studied and deployed the OSOAD approach in developing an online agent-based trading and mining support system – F-Trade. Some main functions available for supporting trading and mining in the F-Trade include: plug-and-play, data gateway, profiles and business-oriented interaction, online system customization and reconstruction, optimization of trading strategies, stock-strategy pairs, supporting comprehensive add-on applications from capital markets, etc.

The F-Trade is currently running on three superservers, including the main application server and local database server located at Faculty of IT, University of Technology, Sydney at Broadway, and remote stock data warehouse of AC3 [1] at Australian Technology Park at Redfern Railway Station. More than 20 trading and mining algorithms have been plugged into the F-Trade, and tested via connecting to real stock data at CMCRC, CSFB. It was also demonstrated to international conferences such as IAT04 [16] and PAKDD [20].

In [3], we have reported the results of functional evaluation, nonfunctional evaluation, and empirical evaluation. Our research and experiment in this work have shown that OSOAD approach is effective and flexible for engineering an open complex agent-based system, with capacities of interoperability, adaptability, user-friendliness, and privacy keeping.

8. Related Work and Discussion

A number of OOM-related AOSE approaches, for instance, GAIA, MASE, MESSAGE, OMNI, ROADMAP, SODA, TROPOS, etc., to some degree implicitly embody or explicitly embody the OOM. We compared the above approaches with our OSOAD in two aspects: (i) what system attributes captured by the ORGANISED framework are supported, (ii) which software engineering phases are covered.

Table 6 summarizes the above comparison in terms of main attributes in an OCAS such as O (organization), G (goal), A (actor and role), N (norm), I (interaction), R (rule), E (environment), S (Structure), D (dynamics), and modeling techniques such as V (visual modeling) and F (formal modeling). Compared with the above existing OOM-related approaches, we find that our OSOAD approach following the ORGANISED framework demonstrates promising for explicitly analyzing all main attributes in open agent systems both visually and formally.

Table 6. Comparison of organization-oriented analysis

	O	G	A	N	I	R	E	S	D	V	F
GAIA	√	√	√	√	√	√	√	√		√	√
MASE		√	√		√			√		√	
MESSAGE	√		√		√			√		√	
OMNI	√	√	√	√	√	√		√			√
OSOAD	√	√	√	√	√	√	√	√	√	√	√
ROADMAP	√	√	√	√	√	√	√	√	√	√	√
SODA	√		√		√	√	√	√			
TROPOS	√	√	√		√	√		√		√	√

Table 7 further compares the above mentioned approaches in terms of main software engineering phases: ERA (early requirement analysis), LRA (late requirements analysis), AD (architectural design), DD (detailed design), and I (implementation). We can further find out that all the above five phases can be supported in OSOAD.

Table 7. Comparison of software engineering phases

	ERA	LRA	AD	DD	I
GAIA		√	√	√	
MaSE		√	√	√	
MESSAGE		√	√	√	
OMNI		√	√		
OSOAD	√	√	√	√	√
ROADMAP	√	√	√	√	
SODA		√	√		
TROPOS	√	√	√	√	√

The above comparison shows that the OSOAD is a systematic approach for engineering OCAS. It can not only capture and abstract all essential organizational members in an agent organization, but also supports complete and detailed analysis, design and implementation of the system. In addition, the transition from organizational abstraction to system design and implementation can be undertaken smoothly.

9. CONCLUSIONS AND FUTURE WORK

The OSOAD approach aims to engineer OCAS. It donates detailed and deployable mechanisms for organizational abstraction, organization-oriented analysis, agent service-oriented design, and Java agent service-based implementation.

The OSOAD is, to the best of our knowledge, the first work in the world which integrates organization-oriented modeling and service-oriented computing. This integration has shown to be effective in analyzing and designing open agent systems. The dialogue between OOA and ASOD is based on the insight that the service-oriented computing is most suitable for designing large-scale distributed software organization. The successful interaction between OOA and ASOD presents a promising solution for effective representation of system design, and smooth transition from organization-oriented analysis to design. The OSOAD embeds explicit, relatively complete and deployable integration methodology and model-building blocks to support smooth transition from OA, OOA to ASOD, and practical implementation based on multiagent-driven service systems in Java.

On the basis of the current progress of the OSOAD, there are following key aspects which are under development towards a systematic organization-oriented software engineering.

- Improvement and enhancement of organization-oriented abstraction
- Improvement and enhancement of organization-oriented analysis
- Development of agent services-oriented design and implementation
- Seamless bridge from organizational abstraction to analysis, design and implementation
- Operation and management of agent services

REFERENCES

[1] AC3: www.ac3.com.au

[2] G. Caire, et al. "Agent-oriented analysis using message/uml", *Lecture Notes in Computer Science*, *vol. 2222*. Springer Verlag, New York, pp.119–135, 2002.

[3] L.B. Cao. Organization and service oriented analysis and design – engineering open complex agent systems. *Technical Report*, University of Technology Sydney, Australia, 2004.

[4] L.B. Cao, R.W. Dai. "Autonomous Intelligent Agents-based Metasynthetic Engineering: A Macroeconomic Decision-Support System", First International Congress on Autonomous Intelligent Systems(ICAIS2002), ICSC Academic Press, pp60-66, Canada/The Netherlands, 2002.

[5] L.B. Cao, R.W. Dai. Social Abstraction for Agent-based Open Giant Intelligent Systems, Proceedings of International Conference on Intelligent Information Technology (ICIIT-02), pp.47-52, (ISBN 7-115-75100-5/0267)

[6] L.B. Cao, J.Q. Wang, L. Lin, and C.Q. zhang. Agent Services-Based Infrastructure for Online Assessment of Trading Strategies. *Proceedings of the 2004 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, IEEE Computer Society Press, 345-349.

[7] K.M. Carley. "Computational and Mathematical Organization Theory: Perspective and Directions", *Computational and Mathematical Organization Theory*. 1(1): 39-56, 1995.

[8] K.M. Carley: Artificial Social Agents. http://www.hss.cmu.edu/departments/sds /faculty/carley/publications.htm, 2001

[9] CMCRC: www.cmcrc.com

[10] M. Dastani, et al.: Issues in multiagent system development, In Proceeding of AAMAS2004, pp.922-929.

[11] V. Dignum, J. Vazquez-Salceda, F. Dignum. A Model of Almost Everything: Norms, Structure and Ontologies in Agent Organizations. AAMAS04

[12] T. Erl: Service-Oriented Architecture : A Field Guide to Integrating XML and Web Services. Pearson Education, 2004

[13] F-Trade: http://datamining.it.uts.edu.au:8080/tsap

[14] A. Garcia et al. (eds) Software engineering for large-scale multi-agent systems. Springer, 2003.

[15] F. Giunchiglia, J. Mylopoulos, A. Perini. The Tropos Software Development Methodology: Processes, Models and Diagrams. AOSE02

[16] IAT04: http://www.maebashi-it.org/IAT04/

[17] T. Juan, A. Pearce, L. Sterling. ROADMAP: Extending the GAIA Methodology for Complex Open Systems, AAMAS02

[18] J.J. Odell, H.V.D. Parunak, M. Fleischer, S. Brueckner. Modeling agents and their environment. *AOSE2002*

[19] A. Omicini. SODA: Societies and Infrastructures in the Analysis and Design of Agent-based Systems. In AOSE'2000, Limerick (Ireland), June 10 2000.

[20] PAKDD04: http://www.deakin.edu.au/~pakdd04/

[21] X.S. Qian. Building systemtology, Shanxi Science and Technology Press, 2004

[22] A. Vasilyev. Synergetic Approach in Adaptive Systems. Master thesis, Transport and Telecommunication Institute, Riga, Latvia, 2002

[23] F. Zambonelli, N.R. Jennings and M. Wooldridge. "Developing multiagent systems: the GAIA Methodology", *ACM Trans on Software Engineering and Methodology*, 12(3):317-370, 2003

[24] M. Wood, S.A. Deloach, C. Sparkman. "Multiagent system engineering", *Int. J. Softw. Eng. Knowl. Eng*. 11(3): 231–258, 2001.