

The OSOAD Methodology for Open Complex Agent Systems

Longbing CAO, Chengqi ZHANG and Ruwei DAI

Abstract—Open complex agent systems (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 a 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 the 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 the practical development of agent service-based financial trading and mining applications.

Index Terms—Open complex agent system, organization oriented modeling, service-oriented computing, agent-based computing

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 [13]. 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 one of most fundamental tasks; 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 [8] and artificial social systems [9] to seek original and effective supports. 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 [21], MESSAGE [2], OperA [11], ROADMAP [16], SODA [17], etc.

However, there exist many issues in existing OOM-related methodologies and approaches [11, 3]. For instance, there is 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 the existing approaches.

In more than six years of work on developing an agent-based macro-economy decision support system [4, 5], and 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) [10], we targeted and built a practical organization-oriented AOSE approach and its model-building blocks. 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). 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 agent-based computing, organization-oriented modeling and service-oriented computing. The dialogue between them leads to bilateral benefits for the abstraction and the modeling on the basis of 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 methodology 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 a 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,

Manuscript received on Jan 15, 2005; revised on May 25, 2005. This work was supported by Capital Market CRC Australia for the Data Mining Program (www.cmcrc.com), UTS Research Fellowship Funding and data supports from AC3 Australia.

Longbing CAO and Chengqi ZHANG are with Faculty of IT, University of Technology, Sydney (e-mail: lbcao@it.uts.edu.au and chengqi@it.uts.edu.au); Ruwei DAI is with Institute of Automation, Chinese Academy of Sciences (e-mail: ruwei.dai@ia.ac.cn).

openness, distribution, interaction, hierarchy, integration, socialization, uncertainty and human involvement. For instance, in an OCAS, goals would be multiform and hierarchical; its environment is often open, networked, heterogeneous and uncertain; the interactions, agent activities and behaviors often take place temporally, spatially or spatial-temporally; the constraints and the rules on social interactions may be presented in terms of multiple dimensions; in addition, the system structure would be rather dynamic and complicated.

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

1) Most of the 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) Each of the OOM phases hasn't been investigated clearly enough. For instance, some phases are not supported, architectural and detailed designs are merged.

4) None of the 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. In practice, formal specifications and integrative modeling embedding both visual and formal specifications are essential for the complete and precise analysis of OCAS.

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 major system attributes in 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 organization and organizational theory [8, 9]. Furthermore, the modeling of OCAS can also benefit from other disciplines such as system sciences and the science of complexity [20] for deeply 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 thoughts and the investigation of the existing OOM-related AOSE approaches [3], we find out that there is big disagreement among the existing approaches in aspects such as major system attributes, modeling concepts and techniques. More importantly, some important system attributes, for instance, system dynamics, are not covered by almost all of the existing approaches. Therefore, we propose a new framework, referred to as *ORGANISED*, which targets a unified and relatively complete organization-oriented view of OCAS.

The *ORGANISED* framework is composed 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 the *actor* available from the existing approaches. In particular, the meanings and ranges of its 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 fundamental attributes in OCAS, have been highlighted in this framework. Figure 1 shows the metamodel of the *ORGANISED* framework.

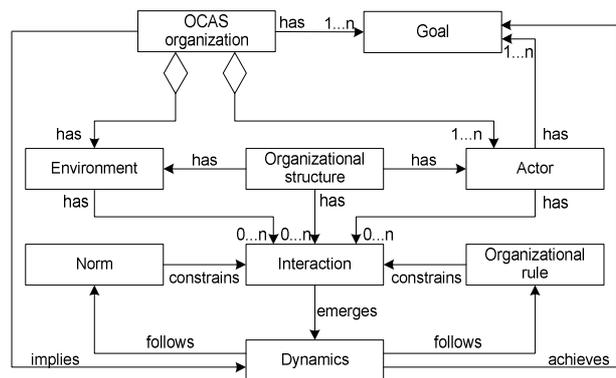


Fig. 1. The *ORGANISED* metamodel

3.2 The OSOAD Methodology

The OSOAD methodology embeds 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 system implementation. The above are packed together and called *Organization and Service Oriented Analysis and Design* (i.e. OSOAD methodology).

- 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 its problem-solving system. In addition, the

ORGANISED framework is recommended to capture all major 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, (viii) integration level and strategy, and so on.

- 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, (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 F-Trade [13], the following briefly lists hybrid policy and philosophy in capturing the ORGANISED members, the 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 the progressive refinement of the decomposition and the aggregation first, and finally form the overall integrated ORGANISED framework, 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 OOA, the following aspects are specified. A simplified OOA activity list is shown in Table 1.

- 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 that the system must hold;

- Presenting the above two types of requirements in diagrammatic notations and/or formal specifications;

- Analyzing the evolutionary activities and its process of a 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 1. 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 [7]. 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.

A set of techniques have potential for analyzing organizational dynamics. Here we take the POMDP_{AEI} model [19] as an example. In 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 2 and 3, 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 ($t_1 < t_2$).

- $\exists apr: \text{AlgoPluginRequest} (apr.\text{depender} = \text{PLUGINPERSON} \wedge apr.\text{dependee} = \text{PluginInterface} \wedge \text{Fulfilled}()) \rightarrow [\exists \leq t_2 \text{CheckAlgorithmValidity}] \exists \leq t_1 \text{AcceptPluginRequest.Fulfilled}()$

Table 2. *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 3. *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)

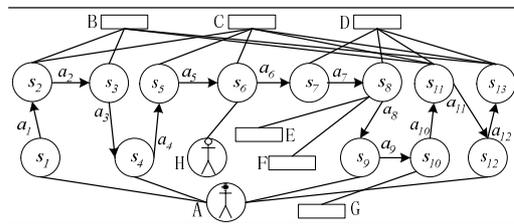


Fig.2. POMDP state chain for *AlgoConfigureAgent*

4.3 Scenario-Based Sequential Analysis

In OOA, it is often important to refine the analysis to get the right requirements. To this end, *scenario-based sequential analysis* is a good technique to work through the refinement of analyzing early and late requirements. Scenario-based sequential analysis is a sequential and gradual process, in which we can investigate every possible scenario and assist the refinement and manage conflicts in the requirements.

In the following, we illustrate the scenario-based sequential analysis through refining the goal *RegisterAlgo* in F-Trade. The objectives of goal *RegisterAlgo* is as follows. When an algorithm component has been coded

and the algorithm isn't available from the system at the moment, the algorithm component can be registered into the system by calling plug-in interfaces, filling in algorithm registration ontologies, and uploading the algorithm module.

The Creation condition for an instance of the goal *RegisterAlgo* is that its predecessor goal *CodeAlgo* has already been fulfilled. The invariant shows constraints on the lifetime of class instances. For the goal *RegisterAlgo*, the invariant binds a *RegisterAlgo* object with its predecessor object. To fulfill the goal *RegisterAlgo*, for each algorithm, there exists only one *AlgorithmComponent* instance that has been coded and will be uploaded at sometime t_2 in the future through calling *CallPluginInterfaces* at future time t_1 ($t_1 \leq t_2$). One legal instance of *FillinAlgoRegisterOntologies* is filled at t_2 . The *FillinAlgoRegisterOntologies* are allowed for multiple instances by following rules in the class *Ontology* for each *AlgorithmComponent*. The above scenario is shown in Figure 3. The creation of an object is indicated by a symbol • in the diagram. The states of creation and fulfillment of relevant objects are shown on the sequential scenario diagram as well.

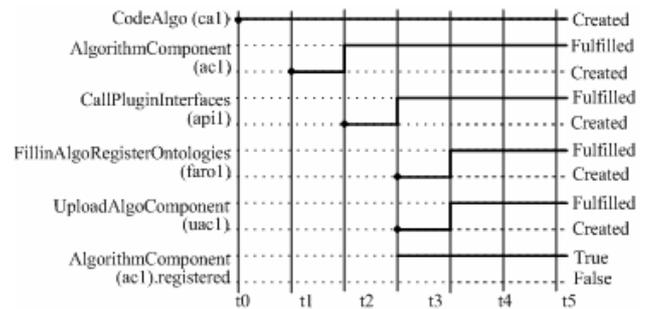


Fig. 3. A sequential scenario for *provider* to register algorithm

As shown in the above small case study, formal modeling can benefit responsibility refinement and conflicts management through scenario-based analysis in the process of OOA.

5. AGENT SERVICE ORIENTED DESIGN

The *agent service-oriented design* is composed of two levels of system design namely *agent service-oriented architectural design* and *agent service-oriented detailed design*. Notice that this process is iterative as recommend 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 are highlighted.

- Agent Service Abstract Model: This defines the types of agent and service architecture, the properties and attributes of agent services.

- **Agent Service Design Pattern:** It abstracts some high-level agent architectural frameworks, 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 the management and communications of agent services include the naming, directory, communication, transport, mediation and discovery of agent services in integrated enterprise applications.

For instance, in order to integrate some legacy applications, we develop agent services such as proxy, wrapper and adapter for atomic transactions, gateway for data access, 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. There are two applications. 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 center server through business services. In addition, there are another two applications 3 and 4 are combined with the server through 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 local 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 the mechanisms for the directory, communication, transport, mediation and discovery of agent services.

- **Agent Service-Oriented Design Strategies:** Here some strategic issues in designing agent service-oriented systems 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 the endpoint interface. 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 4, in which the operation Implementing Algorithm (ImplementAlgo) is performed by methods Implementing Algorithm API (ImplementAlgoAPI) and Coding Algorithm Logics (CodeAlgoLogic).

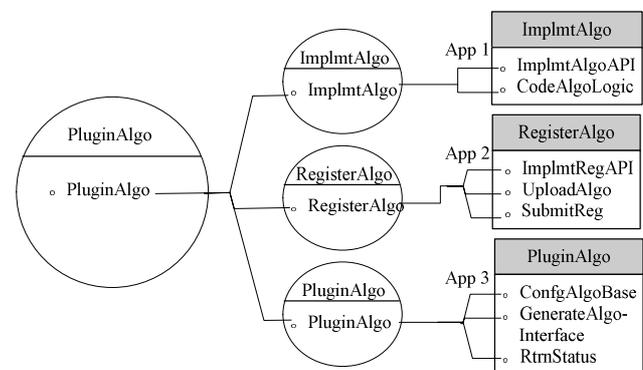


Fig. 4. An agent service with multiple operations via combining multiple agents' methods

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

6. INTERACTION BETWEEN OOA AND ASOD

After building the mechanisms 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 do not discuss it for space limitation; detailed information will be 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 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 4 lists the mapping between some elements in OOA and their corresponding items in ASOD as discussed.

Table 4. 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 use of the second-generation Web services.

The basic work for the Web service-driven agent infrastructure includes the following points.

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

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

- Agentized components: applications and application components in enterprise application integration can be agentized on demand for establishing some specific functional components. For instance, agents for human-computer interaction, resource access dispatching, the management of Web services, and enterprise business logic. The problem here may include how to link and administer these agents in the Web services-centric environment.

- The management, dispatching, conversation, negotiation, mediation and discovery of Web services: for these issues, agents can play great roles with flexible, intelligent, automated and (pro-) active abilities. For instance, a gateway agent located at a remote data source can listen 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.

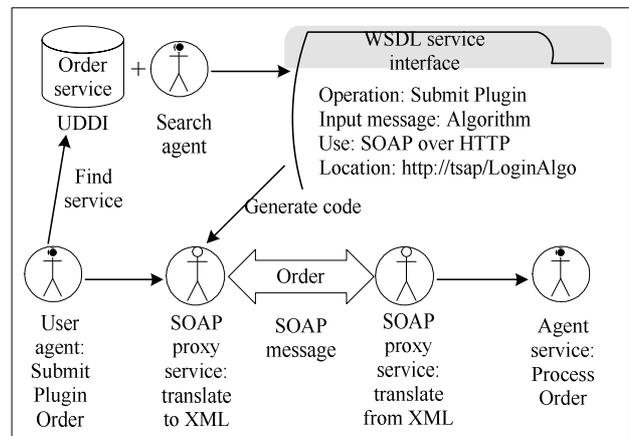


Fig. 5. Web services-driven agent systems for EAI

Figure 5 describes the architecture of a Web services-driven agent system. In this system, a client defines and submits the algorithm *Plugin Order* by 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 agent-based computing and service-oriented computing. Service-oriented computing is not necessary to be Web service-driven or -centric. In this approach, agents act as the main building blocks of a system. A system following this approach is called “*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, 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 in our work [3].

7.2. Case Study and Evaluation

We studied and deployed the OSOAD methodology in developing an online agent-based trading and mining support system – F-Trade. Some major functionalities available for supporting trading and mining in 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 super-servers, 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 by connecting to stock data at AC3 and CMCRC.

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 the OSOAD approach has potential for effectively and flexibly 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, OperA, ROADMAP, SODA, TROPOS, to some degree implicitly or explicitly embody the idea of OOM. We compared the above approaches with our OSOAD in two aspects: (i) what system attributes captured by the ORGANISED framework are supported by them, (ii) which software engineering phases are covered in them.

Table 5 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), as well as 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 promise for explicitly analyzing all main attributes in open agent systems both visually and formally.

Table 5. Comparison of organization-oriented analysis

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

Table 6 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 6. Comparison of software engineering phases

	ERA	LRA	AD	DD	I
GAIA		√	√	√	
MaSE		√	√	√	
MESSAGE		√	√	√	
OperA	√		√	√	
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 methodology 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 agent-based computing, 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 an effective representation of system design, a smooth transition from organization-oriented analysis to design. The OSOAD embeds explicit and relatively complete and deployable integration methodology and model-building blocks to support smooth transition from OA, OOA to ASOD, as well as 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 analysis of norm and organizational dynamics;
- Development of organizational structure and design patterns for agent services-oriented design and implementation; and
- Operation and management of agent services.

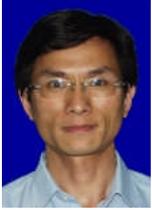
ACKNOWLEDGEMENT

The authors would like to thank the comments provided by the anonymous reviewers and editor, which help the authors improve this paper significantly.

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 and C.Q. Zhang, *Engineering Open Complex Agent Systems*, Springer Verlag, to appear in 2006.
- [4] L.B. Cao and 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, 2002.
- [5] L.B. Cao and 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, 2002.
- [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, 2004.
- [7] Cao, L.B. and Zhang, C.Q. and R.W. Dai, "Organization-oriented analysis of open complex agent systems," *Int. J. on Intelligent Control and Systems*, Vol.10. No.4, 2005.
- [8] K.M. Carley. "Computational and Mathematical Organization Theory: Perspective and Directions," *Computational and Mathematical Organization Theory*, 1(1): 39-56, 1995.
- [9] K.M. Carley, "Artificial Social Agents," <http://www.hss.cmu.edu/departments/sds/faculty/carley/publications.htm>, 2001.
- [10] CMCRC: www.cmcrc.com.
- [11] M. Dastani, et al., "Issues in multiagent system development," In *Proceeding of AAMAS2004*, pp.922-929, 2004.
- [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 and A. Perini, "The Tropos Software Development Methodology: Processes, Models and Diagrams," *Technical Report No. 0111-20, ITC - IRST*, Nov 2001.
- [16] T. Juan, A. Pearce and L. Sterling, "ROADMAP: Extending the GAIA Methodology for Complex Open Systems," *Proceedings of AAMAS02*, ACM, pp3-10.
- [17] A. Omicini, "SODA: Societies and Infrastructures in the Analysis and Design of Agent-based Systems," *Proceedings of AOSE'2000*, pp185-193, 2000.
- [18] X.S. Qian, *Building Systemtology*, Shanxi Science and Technology Press, 2004.
- [19] A. Vasilyev, *Synergetic Approach in Adaptive Systems*. Master thesis, Transport and Telecommunication Institute, Riga, Latvia, 2002.
- [20] M. M. Waldrop, *Complexity: the Emerging Science at the edge of Order and Chaos*. Simon & Schuster, 1992.
- [21] M. Wood, S.A. Deloach and C. Sparkman. "Multiagent system engineering", *Int. J. Softw. Eng. Knowl. Eng.* 11(3): 231-258, 2001.
- [22] 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.

Longbing Cao received Ph.D. degrees in computer science in 2002 from Chinese Academy of Sciences, China.



He is a Research Fellow of the University of Technology, Sydney, Australia. He was a Chief Technical Officer from 2002 to 2004 at Guoxin Intelligent, China. He has published more than 40 papers, four books on complex systems and intelligence science, multiagent technology and data mining.

He served as a program committee member on more than 10 Int'l Conferences such as AAMAS04 and AAMAS05. Dr. Cao is a member of IEEE.



Chengqi Zhang received a PhD degree from the University of Queensland, Brisbane in Computer Science and a Doctor of Science (higher doctorate) degree from Deakin University.

He is currently a research professor in Faculty of Information Technology at University of Technology, Sydney.

He has published more than 200 refereed papers, edited nine books, and published three monographs.

He is a Senior Member of the IEEE Computer Society (IEEE), an Associate Editor or a member of the editorial board for five international journals. He has served as General Chair, PC Chair, or Organising Chair for four international Conferences and a member of Program Committees for many international or national conferences.



Ruwei Dai received a B.Sc. degree in Beijing University in 1955. He was elected to a member of Chinese Academy of Sciences (CAS) in 1991.

His research interests are artificial intelligence, Chinese character recognition and systems sciences. He has published at least 5 books and 220 articles.

He is a professor of Institute of Automation, the president of Chinese Association of Automation, deputy director of department of engineering science of CAS, academic committee chairman of Sino-Canadian high-tech Center of Resources and Environment, and is the chief editor of Chinese Journal Pattern Recognition and Artificial Intelligence.