INTERNATIONAL JOURNAL OF INTELLIGENT CONTROL AND SYSTEMS
VOL. 10, NO. 2, JUNE 2005, 114-122

Organization-Oriented Analysis of Open Complex Agent Systems

Longbing CAO, Chengqi ZHANG and Ruwei DAI

Abstract- **Organization-oriented analysis acts as the key step and foundation in building organization-oriented methodology (OOM) to engineer multi-agent systems especially open complex agent systems (OCAS). A number of existing approaches target OOM, while they are incompatible with each other, and none of them is available as a solid and practical tool for engineering OCAS. This paper summarizes our investigation in building a unified framework for abstracting and analyzing OCAS organizations. Our organization-oriented framework, referred to as ORGANISED, integrating and expanding existing approaches, explicitly captures the main attributes in an OCAS. Following this framework, individual model-building blocks are developed for all ORGANISED members; both visual and formal specifications are utilized to present an intuitive and precise analysis. The above techniques have been deployed in developing an agent service-based trading and mining support infrastructure.**

***Index Terms*—Open complex agent system, organization oriented analysis, ORGANISED framework**

1. INTRODUCTION

Middle-size and large-scale open agent systems [8], referred to as *open complex agent systems* (OCAS) [3], play increasingly important roles in complex problem solving, such as online automatic trading. Engineering such systems is a major yet challenging area in agent-related research. Agent-centric organization-oriented analysis, design and implementation, namely *organization-oriented methodology* (OOM), has emerged as the highly promising foundation for *agent-oriented software engineering* (AOSE) of OCAS [3]. A number of AOSE researchers have referred organizational metaphor and theory [6] to seek original and effective support for developing organization-oriented AOSE, such as Formal TROPOS [9], GAIA [19], MASE [20], MESSAGE [2], ROADMAP [10], SODA [15], and the like.

The above studies, to varied degrees, adopt ideas from human organizations and organizational theory towards agent-centric OOM. As we have seen, most of existing work is still preliminary, and is incompatible with each other in concepts and modeling techniques. None of them is available as a solid and practical tool for engineering OCAS. On the other hand, some existing work, such as Formal TROPOS and GAIA, does bring in good concepts and techniques for handling some aspects of OOM. It is worthy of studying to analyze their strengths and weaknesses, and integrate and expand them to build an effective unified mechanism for modeling OCAS-like agent organization.

This paper reports our work in investigating, integrating and expanding existing OOM-related approaches, and targets a unified and actionable framework for *organization-oriented analysis* (OOA) since it is the groundwork of OOM[1]. However, this does not mean a simple addition of existing techniques; rather the OOA is constructed with the following principles in mind.

- Developing a new framework for OOA, which captures almost all main attributes in an agent OCAS-like organization.
- Selecting, integrating and expanding good concepts and techniques from the existing knowledge base on demand to instantiate the proposed framework and its related members.
- Building new modeling techniques for remaining members in the framework.
- Presenting the OOA framework in specifications for concrete and precise analysis.

The OOA framework is called *ORGANISED* for explicitly capturing major attributes in an agent organization. In light of this framework, model-building blocks for all members are developed. Both visual and formal specifications are utilized to present these building blocks in an actionable way. We exemplify this approach with an online agent-based system F-Trade (Financial Trading Rules Automated Development and Evaluation) [4]. It shows that the ORGANISED framework is effective in analyzing main attributes in an OCAS.

This paper is organized as follows. Section 2 briefly introduces the case study system – F-Trade by which we exemplify the OOA modeling blocks. The ORGANISED framework and the OOA process are outlined in Section 3 and 4, respectively. In Section 5, the ORGANISED members in the OOA are modeled both visually and formally. Section 6 discusses the related work. Finally, Section 7 summarizes and presents our future work.

2. CASE STUDY SYSTEM – F-TRADE

F-Trade[2] is an agent service-based automated enterprise infrastructure [4] for the back-testing, simulation,

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).

[1] For organization-oriented design, we have built a system called *agent service-oriented design* (ASOD) [3] by integrating multi-agent and service-oriented computing.

[2] The current version F-TRADE 2.0 is accessible by http://datamining.it.uts.edu.au:8080/tsap.

evaluation and optimization of trading strategies and data mining algorithms with online connection to huge amounts of stock data.

The main objectives in building the F-Trade are to provide financial traders and researchers and data miners with a flexibly and automatically practical infrastructure. With this infrastructure, they can plug in their algorithms easily, and concentrate on improving the performance of their algorithms with real and iterative evaluation on a large amount of real stock data from international markets. All other work, including user interface generation, data preparation, and resulting output is maintained by F-Trade. For financial traders, for instance, brokers and retailers, the F-Trade presents them a real test bed, which can help them take no risk to iteratively evaluate their favorite trading strategies before they put money into the real markets. On the other hand, the F-Trade presents huge real data in multiple international markets, which is used for both realistic back-testing and simulation of trading strategies, and the optimization of trading strategies using data mining techniques.

The F-Trade looks like an online services provider. Figure 1 shows its architecture. As a systematic infrastructure supporting data mining, trading evaluation and finance-oriented applications, the F-Trade encompasses comprehensive functions and services. They are categorized into the following groups: (i) trading services, (ii) mining services, (iii) data services, (iv) algorithm services, and (v) system services.

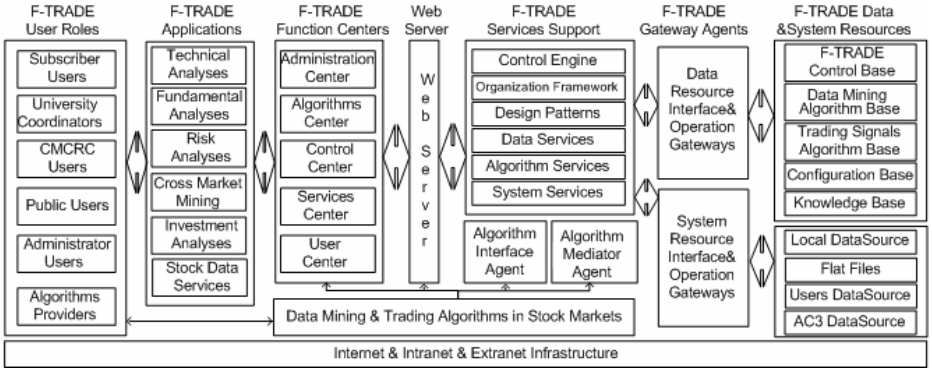


Fig. 1 . The architecture of F-Tra de

The F-Trade is very flexible. More than 20 practical trading strategies have been developed and plugged into the F-Trade via the soft plug-and-play [5]. Both research and applications such as new trading and mining algorithms on capital markets, multiple data sources, system modules for technical analysis, fundamental analysis, investment decision support and risk management can be easily embedded into the F-Trade. The system has been running online for three years very robustly.

3. The ORGANISED FRAMEWORK

This section briefly introduces the ORGANISED framework and the process undertaking organization-oriented analysis of an OCAS-like organization.

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 [6]. Furthermore, the modeling of OCAS can also benefit from other multiple disciplines such as system sciences and the science of complexity [18] 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 varied 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 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.

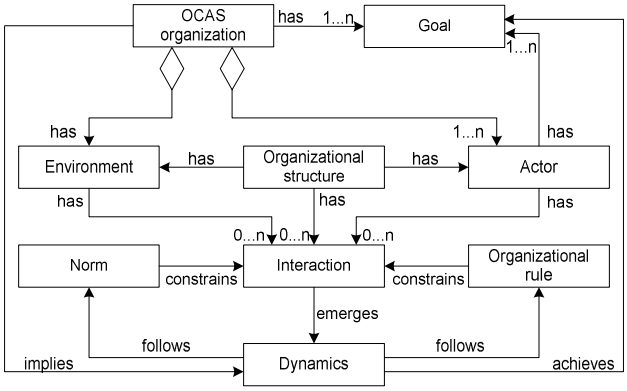


Fig. 2. The ORGANISED metamodel

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 actors available from the existing approaches, in particular the meanings and ranges of some members have been expanded. For instance, in our framework, an actor may take a 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 2 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
Structure	Organizational structure is a collectively emergent architecture-oriented pattern from the interaction among system members at the corresponding system level following certain rules
Environment	Environment is a relative object that may comprise actors, principles, processes and forces [14] 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, discrete-event dynamic, or continuous-time dynamic system following patterns such as self-organizing, center-controlled and stochastic bodies

4. ORGANIZATION-ORIENTED ANALYSIS PROCESS

The aim of the ORGANISED is to provide an easily understandable and actionable framework for analyzing complex agent organizations. In [3], we have presented a detailed description for implementing the process of OOA following the ORGANISED framework. A simplified OOA activity list is shown in Table 2.

Table 2. Key activities and questions in 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 the functionalities of problem-solving system.
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 implement?
Prioritize requirements	What are the most important goals and interaction activities the system must do?
Prototype for feasibility and discovery	Can the proposed organization-oriented model-building technology 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 with management	Should we continue to design and implement the system we propose?

In addition, according to our empirical practice in the development of F-Trade, the following lists hybrid philosophies [3] for capturing the ORGANISED members in an OCAS.

- Reductionism for top-down decomposition – in decomposition, the reductionism philosophy is taken. 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 or the system.
- Systematology [16] 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 via taking the philosophy of systematology.

In the next section, we briefly introduce the modeling of the ORGANISED members through illustrating some objects in F-Trade. At this stage, norm is merged into organizational rule and will be investigated as future work. In addition, the actor is introduced first because it is the basic attribute of an agent organization and widely used in modeling other members.

5. MODELING the ORGANISED MEMBERS

In the OOA, rules, goals, actors, interaction, structure and environment are instantiated into organizational rule model, goal model, actor model, interaction model, organizational structure model, and environment model, respectively. System dynamics is discussed individually. In the modeling, good techniques from TROPOS and GAIA are synthesized. Besides diagrammatic notations for these models, formal specifications based on temporal logic [15] are also presented to complement visual modeling. Formal specifications are based on the following operators:

○/● - next/previous state
◊/♦ - sometime in the future/past
□/■ - always in the future/past
$\Diamond_{\leq d}$ (some time in the future within deadline *d*)
$\Box_{\leq d}$ (always in the future up to deadline *d*)
x^* - *x* occurs 0 or more times
x^+ - *x* occurs 1 or more times
x.y - *x is* followed by *y*
x|*y* - *x* or *y* occurs
x∋*y* - *y* is covered by *x*
$[x]_y$ - *x* happens if *y* occurs

5.1. Actor Model

The following types of actors may be identified and presented visually in an agent organization.

- Human Actor (人): human beings related to the system, e.g., algorithm providers.
- Workspace Actor (◯): it universally refers to the proposed system or problem space, for instance, the F-Trade.
- Autonomous Actor (Ⓐ): they are usually agents who fulfill some responsibilities with decision controlled by themselves, e.g., algorithm plug in agent.
- Service Actor (Ⓢ): it performs some activities or functions associated with certain autonomous actors or on its own, e.g., stock data service.
- Resource (□): varying databases, knowledge bases, configuration files, etc.

Every actor has some associated attributes, creation and inner properties (property is labeled as Ⓟ). Agent and Service actors may hold mental states such as belief, desire and intention (state as Ⓢ). Some actors take on roles (role as Ⓡ); one actor may take on one or many roles. The following excerpt example formally illustrates an actor named AlgoPluginAgent that holds the Role PLUGINPERSON to register an algorithm into the F-Trade (the role model follows the GAIA role schema).

Role PLUGINPERSON
InformalDef *This role involves applying for registering a nonexistent algorithm, typing in attribute items of the algorithm, and submitting plug in request to F-Trade.*
FormalDef
Agent AlgoPluginAgent
Attribute constant algoid: AlgorithmID
Existed: Boolean
Attribute constant ai: AttributeItems
Attribute constant apa: AlgoPluginAgent
Protocol CheckAlgorithmValidity
Protocol SubmitAlgoPluginRequest
Permissions
$\forall$ ia: InterfaceAgent(apa.Call(ia) $\rightarrow$ CallPluginInterfaces()) $\rightarrow \Diamond_{\leq t1}$ apa.Read(algoid) $\rightarrow \Diamond_{\leq t2}$ apa.Fill(ai)
Responsibilities
Liveness:
$\exists$ ApplyRegistration.Fulfilled() $\rightarrow$ $\Diamond_{\leq t1}$ CheckAlgorithmValidity.Fulfilled() $\rightarrow$ $\Diamond_{\leq t2}$ FillinAlgoRegisterOntologies.Filled() $\rightarrow \Diamond_{\leq t3}$ SubmitAlgoPluginRequest.Fulfilled()
Safety (Invariant):
$0 < t_1 < t_2 < t_3$

5.2. Modeling Organizational Rules

There are two types of fundamental organizational rules. Structural rules identify the organizational relations among actors and roles; while problem-solving rules specify how an organization solves its problems.

5.2.1. Structural rules

The following list defines four main structural rules in an agent organization and their notations in our modeling. The symbols A and B represent actors, roles or goals.

- Control: B is controlled by A, or B can be achieved by means of A
- Peer: A and B share a peer-to-peer relation
- Ownership: A owns B
- Dependency: A depends on B

Some of the above relationships normally co-exist in some combinatory manner in a complex system. For instance, the following points illustrate four types of combinations of Dependency relationships in TROPOS. In Figure 5 relationship combination is also demonstrated.

- Single Dependency: A solely depends on B
- Bidirectional Dependencies: A depends on B for some situation, for other situation B depends on A
- And/Or Composition: A, B and C depend on D for one and/or multiple dependencies respectively, and;

• And/Or Decomposition: A depends on B, C and D for one and/or multiple dependencies respectively.

The above structural rule relationships are represented by symbols as shown in Figure 3.

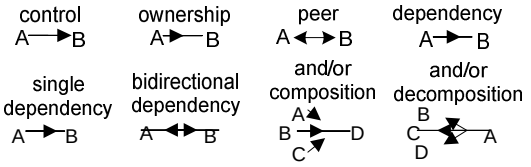


Fig. 3. Symbols representing structural rules.

Relationships can also be presented formally. For instance, the following grammar specifies the relationship Dependency.

/ Relationship grammar*/*
<*relationships*> ::= <*type*> <*relationship*>
<*relationship_type*> ::= (Dependency | Control | Peer | Ownership | …)
<*dependency*> ::= <*type*> Dependency <*name*> <*mode*> Depender <*name*> Dependee <*name*> [*attributes*] [*creation-properties*] [*invar-properties*] [*fulfill-properties*]
<*dependency_type*> ::= (Goal | Subgoal | Resource | [*self-defined*])
<*dependency_mode*> ::= Mode (achieve | cease | maintain | avoid | optimize | …)
<*resource_type*> ::= (Data | Information | Knowledge | [*self-defined*])

5.2.2. Problem-solving rules

Some main problem-solving rules consist of rules for Means-Ends [11], Contribution [11], Goal Decomposition, Iteration and Cardinality. In the following section, we discuss the latter three types of rules.

Goal decomposition [11] defines the refinement of a goal either temporally or spatially. In the process of decomposition, a goal is divided into multiple sub-goals. There are four types of combinations among sub-goals: Sequence, Alternative, Concurrency, and Hybrid.

• Sequence: sub-goals are performed from left to right sequentially to complete the super goal.

• Alternative: a goal can be fulfilled by sub-goal either A or B.

• Concurrency: a goal can only be fulfilled by all decomposed sub-goals in parallel.

• Hybrid: a goal is fulfilled by performing multiple sub-goals in an order combining some of the above three relationships.

Furthermore, a sub-goal may need to be executed iteratively. Iterative links describe under what condition a sub-goal is performed and whether it is done once or repeatedly. There are five types of iterations: While-Loop, For-Loop, Interrupt, If and Pick.

They can be notated by a generic formula as:

[*Notation*] *Predicate* (*ParameterList*).

• While-Loop: **while*(*condition*), reiterate the subgoal while the "condition" is satisfied.

• For-Loop: **for*(*variable, listOfValues*), list of values for the variables in the subgoals are held iteratively

• Interrupt: **whenever*(*variableList, condition*), values for the variables in the subgoals are held whenever the condition is satisfied.

• If: !*if*(*condition*), the sub-goal is operationalized if the condition is satisfied.

• Pick: !*pick*(*variableList, condition*), values for the variables in the sub-goals are picked non-deterministically satisfying the condition.

Moreover, for fulfilling a goal α, one to many goals β may need to be dependent. This refers to Cardinality constraints in an organization. There are four types of cardinalities such as Mandatory One, Mandatory Many, Optional One, and Optional Many.

The above problem-solving rule relationships are represented by symbols as shown in Figure 4.

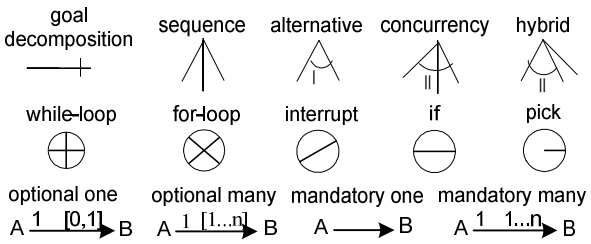


Fig. 4. Symbols representing problem-solving rules.

Figure 5 shows the model of the execution and evaluation of an algorithm in F-Trade by combining some of the above rules.

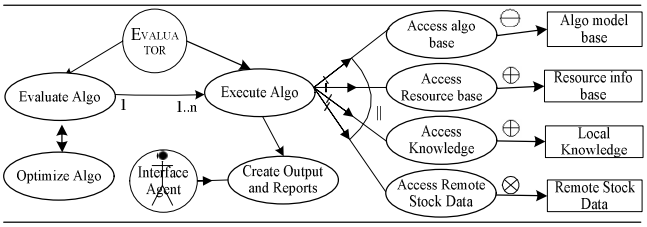


Fig. 5. Combination of organization rules modeling algorithm execution and evaluation.

5.3. Modeling Organizational Goals

Functional (⬭) and nonfunctional (☁) goals [11] can be visualized through goal decomposition in an organizational goal and structure diagram. In addition, in Section 5.5, we introduce another method for building goal diagram based on GAIRE model [3]. Formally, a goal can be represented by a formal grammar. For instance, the goal RegisterAlgo is expressed as follows.

Goal RegisterAlgo
InformalDef *When an algorithm component is coded and the algorithm isn't available from F-Trade, this algorithm can be registered by calling plug-in interfaces, filling in algorithm registration ontologies, and uploading the algorithm configuration base.*
FormalDef
Role AlgoProvider
Agent AlgoPluginAgent
Agent ConfigureAgent
Agent InterfaceAgent
Service OntologyService
Mode achieve
Attribute constant ca: CodeAlgo
Attribute constant aro: AlgoRegisterOntology
Attribute constant algo: Algorithm
registered: boolean
Creation condition
● Fulfilled(ca) ∧ ¬ Existed(algo)
Invariant ca.actor = actor
Fulfillment condition
∀ ac: AlgorithmComponent (ac.algo = algo →
$\Diamond_{\leq t1}$ ∃ cpi: CallPluginInterfaces (cpi.actor = actor ∧ Fulfilled(cpi) ∧ pi.Called) ∧
$\Diamond_{\leq t2}$(∃faro: FillinAlgoRegisterOntologies (faro.depender = actor ∧ Fulfilled(faro) ∧ aro.Filled) ∧ ∃ uac: UploadAlgoComponent (uac.depender = actor ∧ Fulfilled(uac) ∧ ac.uploaded)))

5.4. Interaction Model

Interaction is usually modeled in terms of diagrams such as activity, sequence and state chart in modeling languages like UML and AUML. More specifically, an interaction protocol can link most of the interaction ontologies. Figure 6 shows the interaction protocol ontologies we taken in [3]. Interaction protocol ontologies can be further filled into a protocol diagram [19] and a pattern template [3] on demand.

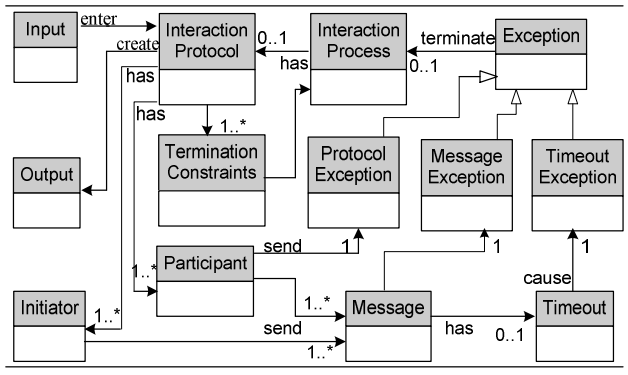


Fig. 6. Interaction protocol ontology

Protocol and pattern can also be formally represented. The following presents the grammar for a protocol.

*/*grammar for protocol*/*
<*protocol*> ::= Protocol <*name*> [*function*] [*Message*] <*initiator*> <*responder*>$^{+}$ <*input*> <*rule*> <*output*> [*termination*] [*exception*]
<*exception*> ::= Exception <*type*>
<*type*> ::= [message] [timeout] [protocol]
*/*grammar for pattern*/*
<*pattern*> ::= Pattern <*name*> [*alias*] <*force*>* [*problem*] <*structure*> [*layer*] <*participant*>$^{+}$ <*solution*> <*dependency*>* <*role*>$^{+}$ [*message*] [*context*] [*known use*] [*example*] [*see also*]

For instance, the protocol SubmitAlgoPluginRequest mentioned in role PLUGINPERSON is expressed as follows.

Protocol SubmitAlgoPluginRequest
Requester: AlgorithmRegisterAgent
Responder: ConfigureAgent
Input: reqid, algoid
Rule:
∀ apa: AlgoPluginAgent(apa. FillinAlgoRegisterOntologies.Fulfilled(algoid) ∧ Fulfilled(algoid)) →
$\Diamond_{\leq t1}$ Fulfilled(reqid)
Output: reqid.Successful()

In addition, messages transferred in an interaction can also be formalized. The following shows the ACL [7] abstract message formula grammar.

/ ACL Message Abstract Formula Grammar*/*
<*formulae*> ::= (<*communicative act*>.<*message parameter*>)*
<*communicative act*> ::= (accept-proposal | agree | cancel | cfp | confirm | disconfirm | failure | inform | inform-if | inform-ref | inform-ref | not-understood | propose | query-if | query-ref | refuse | reject-proposal | request | request-when | request-whenever | subscribe | envelop)
<*message parameter*> ::= ([*sender*] $_{\text{Anonymous}}$ + <*receiver*>$^{+}$ + [reply-to]$^{+}$ + ([*language*] $_{\text{<content> э <language>}}$ + [*ontology*]) . [*content*]$_{\text{<conversation-id> | in-reply-to}}$ + [*encoding*] $_{\text{envelop}}$ + [*protocol*] $_{\text{<conversation-id> | ○ reply-by}}$ + <*conversation-id*> + [reply-with] + [in-reply-to] + [reply-by])

5.5. Modeling Organizational Structure

The basic task for modeling an organizational structure is to build an organizational structure diagram using the modeling components in the previous sections on demand. Since open agent systems are very complicated, it would be difficult to build up and pack all components into one monolithic organizational structure diagram. An alternative method (if suitable) is to develop high-level organizational framework and low-level subsystem organizational structure diagrams, respectively. If the system is complicated enough, multiple hierarchical subsystems need to be decomposed and analyzed. Finally, the low-level diagram for any target subsystem can be built up, which

looks like Fig. 5 but with more details. We do not exemplify such system diagrams here to save space.

In the process of decomposing and modeling a subsystem, we also build a *GAIRE* model [3]. It captures all detailed item-sets of Goals, Actors, Interactions, Rules and Environment (GAIRE for short). For instance, the GAIRE model for the subsystem S_i on layer a is shown as follows.

$G_{ai} = \{g_{i1}, g_{i2}, \ldots, g_{ij}\}$
$A_{ai} = \{a_{i1}, a_{i2}, \ldots, a_{ik}\}$
$I_{ai} = \{i_{i1}, i_{i2}, \ldots, i_{il}\}$
$R_{ai} = \{r_{i1}, r_{i2}, \ldots, r_{im}\}$
$E_{ai} = \{e_{i1}, e_{i2}, \ldots, e_{in}\}$

After we obtain all GAIRE models for every layer, we can build the global organizational framework through two steps. (i) Building GAIRE global structure diagram. Assuming that every subsystem consists of one aggregated goal G_i, actor A_i, interaction I_i, rule R_i and environment E_i, all item-sets of G, A, I, R and E form one GAIRE system. We build a global organizational structure GAIRE diagram by linking all GAIRE item-sets. The global GAIRE diagram, which also looks like the subsystem diagram, assists us with the global framework of an agent organization. (ii) Building global Goal and Interaction diagrams. We further build a global organizational goal diagram G capturing relationships among goal sets: $\{G_1, G_2, \ldots, G_i, \ldots, G_p\}$, and a global interaction model I capturing interaction among actors, goals and environment in subsystem sets: $\{S_1, S_2, \ldots, S_q\}$. The diagrams G and I can be developed based on either a GAIRE system using aggregated item-sets or detailed item-sets on demand. These two diagrams help us obtain the overall framework of organizational goal and interaction dynamics. Figure 7 shows a sample of Goal diagram, which consists of high-level goals and their sub-goals for algorithm plug-in, configuration and execution.

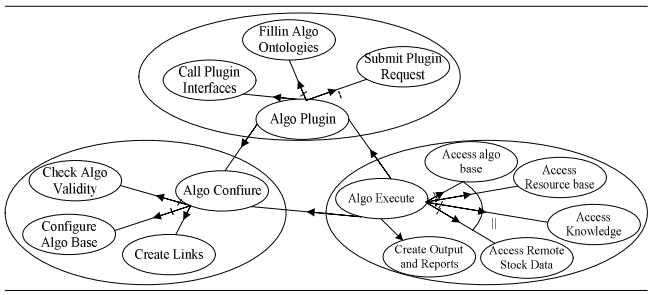


Fig. 7. Goal diagram

5.6. Environment Model

Modeling the environment involves specifying all actors, resources, principles, processes, forces, interactions and the system (or subsystem) boundary. The environment comprises what the organization can exploit, control or consume when it moves towards the achievement of its goals. These are embodied in the environment and structure model.

More specifically, modeling the environment must also cover agent-environment interaction (AEI). By introducing environment observations, an AEI can be further modeled as a Partially Observable Markov Decision Processes (POMDP) [17], called $\text{POMDP}_{\text{AEI}}$. The $\text{POMDP}_{\text{AEI}}$ model is a six-element tuple:

$$\text{POMDP}_{\text{AEI}} = \langle S, A, T, R, O, D \rangle,$$

where $O(s_t, r_t)$ is the finite set of observations related to state s_t and reward r_t; $D(a_t, s_{t+1}, d_{t+1})$ is the probability of making observation d_{t+1} from state s_{t+1} after having taken action a_t. Figure 8 shows the $\text{POMDP}_{\text{AEI}}$ model. In addition, we can also obtain an interaction dynamic diagram as discussed in Section 5.5 based on the GAIRE model. Moreover, environment can be formally modeled with the following grammar.

<*environment*> ::= Environment <*type*> <*actor*>$^+$ [*resources*] [*principles*] [*processes*] [*forces*] [*attributes*] [*invar-properties*]

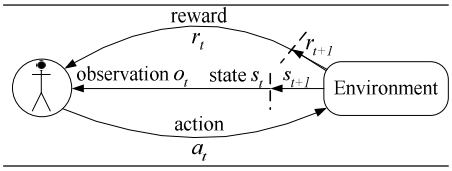


Fig. 8. Agent environment interaction

5.7. Organizational Dynamics Analysis

In literature, the modeling of organizational dynamics is primarily based on methods from the following fields: Markov Decision Process, the Sciences of Complexity [18], Dynamic System Theory [1], logic-based and reactive approaches, and statistical approaches. Additionally, scenarios, sequences, states and activities are often used when system dynamics is visually presented.

In [3], we demonstrate some techniques in analyzing organizational dynamics. Here we take the $\text{POMDP}_{\text{AEI}}$ model as an instance. In F-Trade, the goal of agent *AlgoPluginAgent* is to fulfill the registration of an algorithm into the system. The excerpted corresponding environment states and actions of the agent are listed in Tables 3 and 4, respectively. The state-action chain of this agent interacting with its environment can be modeled as shown in Fig. 9. Furthermore, this state-action chain can be formally specified. For instance, state transfer 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$]_{\Diamond \leq t1}$ 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)

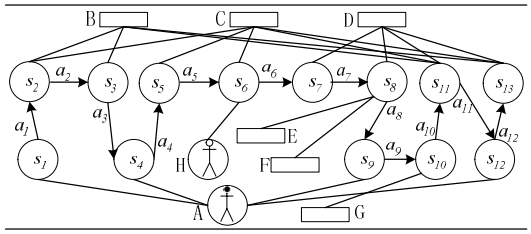


Fig.9. POMDP state chain for *AlgoConfigureAgent*

6. Related Work And Discussion

A number of OOM-related AOSE approaches, for instance, Formal TROPOS, GAIA, MASE, MESSAGE, OMNI, ROADMAP and SODA, to some degree implicitly embody or explicitly adopt the organizational metaphor. Following the attributes captured by the ORGANISED framework, we discuss the characteristics of these approaches from the specific perspective of OOA. GAIA explicitly takes and presents organizational abstraction and late analysis, but nothing related to organizational dynamics. Moreover, OOA support in GAIA, for instance, goal and environment models, and formal representation, are not actionable. MASE does not formally adopt the organizational metaphor other than dealing with goal, interaction and role elements. It uses Use Case for implicit structure analysis. In the MESSAGE, goal support is weak, and no support is provided for rules and environment as in MASE. ROADMAP extends GAIA in aspects such as environment, role, social structures and relationships, but it provides no explicit analysis mechanisms for system dynamics. SODA designs a conceptual framework with limited supports of practical analysis. TROPOS brings in nice techniques for OOM, which mainly depend on goal-oriented analysis, and may be sufficient except that there is no support for environment. Among all of the above approaches, only ROADMAP pays little attention to dynamic changes.

Table 5. Comparison of organization-oriented analysis

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

Table 5 summarizes the above comparison in terms of main attributes in an OCAS such as O (organization), R (rule), G (goal), A (actor and role), N (norm), I (interaction), S (Structure), E (environment), D (dynamics), and modeling techniques such as V (visual modeling) and F (formal modeling). Compared with the above existing OOM-related approaches, the ORGANISED framework demonstrates promise for explicitly analyzing all main attributes in open agent systems both visually and formally.

7. Conclusions And Future Work

The organizational metaphor is effective in analyzing multi-agent systems and in particular open agent systems. However, almost all existing organization-oriented analysis approaches expose intrinsic problems and cannot handle the analysis of OCGS-like agent systems very well. We primarily attribute these problems to inexplicit and incomplete organization-oriented abstraction.

To this end, we have demonstrated an organization-oriented framework, i.e., the ORGANISED, for analyzing OCAS. Corresponding model-building blocks have been developed visually and formally in terms of the ORGANISED members. The main contributions of this paper are as follows:

(i) Proposed an organization-oriented abstraction framework called ORGANISED, which explicitly presents all main members in an OCAS;

(ii) Developed individual and detailed model-building blocks for all ORGANISED members in an OCAS;

(iii) Presented them both visually and formally.

The publicly available trading development and evaluation infrastructure F-Trade has been used as case study to exemplify the above organization-oriented analysis. Our experiments with this system have shown that the proposed approach is helpful and actionable in analyzing open agent systems.

More work is undertaken on modeling norms, model checking and requirement refinement in terms of the proposed framework.

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] Beer, R. D., "A dynamical systems perspective on agent-environment interaction," *Artificial Intelligence*, 72: 173-215, 1995.

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

[3] Cao, L.B. and Zhang, C.Q., *Engineering Open Complex Agent Systems*, Springer Verlag, to appear in 2006.

[4] Cao, L.B., Wang, J.Q. Lin, L. and Zhang, C.Q., "Agent Services-Based Infrastructure for Online Assessment of Trading Strategies," *Proceedings of IAT04*, IEEE Computer Society Press, 345-349.

[5] Cao, L.B., Ni, J.R., Wang, J.Q. and Zhang, C.Q., "Agent Services-Driven Plug and Play in the F-TRADE," G.I. Webb and X.H. Yu (Eds.): AI 2004, *LNAI 3339*, 2004, pp. 917–922.

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

[7] FIPA. *Foundation for Intelligent Physical Agents*, 2004.

[8] Garcia, A., et al. (eds) *Software Engineering for Large-Scale Multi-Agent Systems*. Springer, 2003.

[9] Giunchiglia, F., Mylopoulos, J. and Perini, A., "The Tropos Software Development Methodology: Processes, Models and Diagrams," Technical Report No. 0111-20, ITC - IRST, Nov 2001.

[10] Juan, T., Pearce, A., Sterling L., "ROADMAP: Extending the GAIA Methodology for Complex Open Systems," *Proceedings of AAMAS02*, ACM, pp3-10.

[11] Kolp, M., Giorgini, P., Mylopoulos, J., "A goal-based organizational perspective on multiagent architectures", *LNAI 2333*. Springer-Verlag, New York, 2002, pp.128–140.

[12] Jennings, N.R., "On Agent-Based Software Engineering," *Artificial Intelligence*, 117 (2) 277-296, 2001.

[13] Manna, Z., and Pnueli, A., *The Temporal Logic of Reactive and Concurrent Systems*, Springer-Verlag, 1992.

[14] Odell, J.J., Parunak, H.V.D., Fleischer, M., Brueckner, S., "Modeling agents and their environment: the physical environment," in *Journal of Object Technology*, Vol. 2, No. 2, pp. 43-51, 2003.

[15] Omicini, A., "SODA: Societies and Infrastructures in the Analysis and Design of Agent-based Systems", In *Proceedings of AOSE'2000*, pp185-193, 2000.

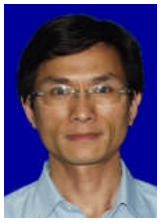
[16] Qian, X.S., *Building Systemtology*, Shanxi Science and Technology Press, 2004.

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

[18] Waldrop, M. M., *Complexity: the Emerging Science at the edge of Order and Chaos*. Simon & Schuster, 1992.

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

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



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.