Organization-Oriented Analysis of Open Complex Agent Systems

Longbing CAO, Chengqi ZHANG, 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 [9], 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 [11], GAIA [21], MASE [22], MESSAGE [2], OMNI [7], ROADMAP [12], SODA [17], 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 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 the strengths and weaknesses of them, and integrate and expand them to build an effective unified mechanism for modeling OCAS-like agent organizations.

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. 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 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 that we’ll 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 related work. Finally, Section 7 summarizes and presents our future work.

2. CASE STUDY SYSTEM — F-TRADE

F-Trade is an agent service-based automated enterprise infrastructure [4] for the back-testing, simulation, evaluation and optimization of trading strategies and data mining algorithms with online connection to huge amount 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.

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 It gets funding support from Capital Market CRC Australia for the Data Mining Program, UTS Research Fellowship Funding and data supports from AC3. The current version F-TRADE 2.0 is accessible by http://datamining.it.uts.edu.au:8080/tasp.
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, etc., 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.

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, and 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 two 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 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 2 further shows the metamodel of the ORGANISED framework.

### Table 1. The ORGANISED framework members

<table>
<thead>
<tr>
<th>Member Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization</td>
<td>An OCAS is an artificial organization; the organization-oriented abstraction and analysis explicitly adopt the organizational metaphor</td>
</tr>
<tr>
<td>Rule</td>
<td>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</td>
</tr>
<tr>
<td>Goal</td>
<td>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 non-functional objectives of an OCAS</td>
</tr>
<tr>
<td>Actor</td>
<td>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</td>
</tr>
<tr>
<td>Norm</td>
<td>Norms may be presented as social patterns governing perceptual, denotive, evaluative, cognitive or behavioral aspects</td>
</tr>
<tr>
<td>Interaction</td>
<td>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.</td>
</tr>
<tr>
<td>Structure</td>
<td>Organizational structure is a collectively emergent architecture-oriented pattern from the interaction among system members at corresponding system level following certain rules</td>
</tr>
</tbody>
</table>
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. 2. The ORGANISED metamodel](image)

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>
<thead>
<tr>
<th>OOA activities</th>
<th>main questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gather information</td>
<td>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?</td>
</tr>
<tr>
<td>Define system requirements</td>
<td>What functional (stakeholders, goals, rules, policies, constraints, etc) and nonfunctional (global qualities of the system) details do we need the system to do?</td>
</tr>
<tr>
<td>Prioritize requirements</td>
<td>What are the most important goals and interaction activities the system must do?</td>
</tr>
<tr>
<td>Prototype for feasibility and discovery</td>
<td>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?</td>
</tr>
<tr>
<td>Generate and evaluate</td>
<td>What’s the best way for organization-oriented analysis to develop the agent-based system?</td>
</tr>
</tbody>
</table>

Table 2. key activities and questions in the OOA

In addition, according to our empirical practice in the 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, 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 [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 via taking the philosophy of systematology.

In the following section, we briefly introduce the modeling of the ORGANISED members through illustrating some objects in the 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 most 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:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>⊕</td>
<td>- next/previous state</td>
</tr>
<tr>
<td>⊕</td>
<td>- sometime in the future/past</td>
</tr>
<tr>
<td>⊥</td>
<td>- always in the future/past</td>
</tr>
<tr>
<td>⊥</td>
<td>- always in the future up to deadline d</td>
</tr>
<tr>
<td>⊥</td>
<td>- sometime in the future within deadline d</td>
</tr>
<tr>
<td>⊥</td>
<td>- always in the future up to deadline d</td>
</tr>
<tr>
<td>⊥</td>
<td>- x occurs 0 or more times</td>
</tr>
<tr>
<td>⊥</td>
<td>- x occurs 1 or more times</td>
</tr>
</tbody>
</table>


5.1. Actor Model

The following types of actors may be identified and presented visually in an agent organization.
- Human Actor (□): 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 that 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).

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 agent organizations 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 shows four types of combinations of Dependency relationships in TROPOS. In Figure 3 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 or multiple dependencies respectively, and;
- And/Or Composition (→): A depends on B, C and D for one or multiple dependencies respectively

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> Dependee <name> Dependee <name> [attributes] [creation-properties] [invar-properties] [fullfill-properties]
<dependency_type> ::= (Goal | Subgoal | Resource | [self-defined])
<dependency_mode> ::= (Data | Information | Knowledge | [self-defined])

5.2.2. Problem-solving rules

Some main problem-solving rules consist of rules for Means-Ends [13], Contribution [13], Goal Decomposition, Iteration and Cardinality. In the following section, we discuss the latter three types of rules.

Goal decomposition (→) [13] 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 (1)**: sub-goals are performed from left to right sequentially to complete the super goal.
- **Alternative (β)**: a goal can be fulfilled by sub-goal either A or B.
- **Concurrency (α)**: a goal can only be fulfilled by all decomposed sub-goals in parallel.
- **Hybrid (γ)**: 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 will be performed and whether it should be done once or repeatedly. There are five types of iterations: the While-Loop, the For-Loop, the Interrupt, the If, and the Pick.

They can be notated by a generic formula as:

\[
\text{Notation} \rightarrow \text{Predicate} (\text{ParameterList}).
\]

- **While-Loop** (\(\oplus\)): *while(condition)*, reiterate the subgoal while the “condition” is satisfied.
- **For-Loop** (\(\otimes\)): *for(variable, listOfValues)*, list of values for the variables in the subgoals will be held iteratively.
- **Interrupt** (\(\ominus\)): *whenever(variableList, condition)*, values for the variables in the subgoals will be held whenever the condition is satisfied.
- **If** (\(\oslash\)): *if(condition)*, the sub-goal will be operationalized if the condition is satisfied.
- **Pick** (\(\odot\)): *pick(variableList, condition)*, values for the variables in the sub-goals will be picked nondeterministically satisfying the condition.

Moreover, for fulfilling a goal \(\alpha\), one to many goals \(\beta\) may need to be dependent. This refers to Cardinality constraints in an organization. There are four types of cardinalities such as Mandatory One (\(\alpha \rightarrow \beta\)), Mandatory Many (\(\alpha \rightarrow Many \beta\)), Optional One (\(\alpha \rightarrow 1\ \beta\)), and Optional Many (\(\alpha \rightarrow Many \beta\)).

Figure 3 shows the model of the execution and evaluation of an algorithm in the F-Trade by combining some of the above rules.

![Figure 3: Combination of organization rules modeling algorithm execution and evaluation](image)

**5.3. Modeling Organizational Goals**

Functional (\(\circ\)) and nonfunctional (\(\bullet\)) goals [13] can be visualized through goal decomposition in an organizational goal and structure diagram. In addition, in Section 5.5, we’ll 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 been coded and the algorithm isn’t available from the 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) \land Exists(algo)**
- **Invariant ca.actor = actor**
- **Fulfillment condition**
  \[\forall ac: \text{AlgorithmComponent} (ac.algo = algo \rightarrow \forall cpi: \text{CallPluginInterfaces} (cpi.actor = actor \land Fulfilled(cpi) \land \text{piCalled}) \land \exists \text{faro: FillinAlgoRegisterOntologies} (\text{faro.dependent} = \text{actor} \land Fulfilled(\text{faro}) \land \text{aro.Filled}) \land \exists \text{uac: UploadAlgoComponent} (\text{uac.depending} = \text{actor} \land Fulfilled(\text{uac}) \land \text{ac.uploaded}))\]

**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 A-UML. More specifically, interaction protocol can link most of the interaction ontologies. Figure 4 shows the interaction protocol ontologies we taken in [3]. Interaction protocol ontologies can be further filled into protocol diagram [21] and pattern template [3] on demand.

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> <responders> <input> <rule> <output>
[termination] [exception]
<exception> ::= Exception <type> <message> [timeout] [protocol]
/*grammar for pattern*/
```
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 Figure 3 but with more details. We do not exemplify such system diagrams here.

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.

\[
G_i = \{g_{i1}, g_{i2}, \ldots, g_{in}\} \\
A_i = \{a_{i1}, a_{i2}, \ldots, a_{im}\} \\
I_i = \{1, 2, \ldots, l_i\} \\
E_i = \{e_{i1}, e_{i2}, \ldots, e_{ip}\}
\]

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 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: \( \{G_1, G_2, \ldots, G_m\} \), and a global interaction model \( I \) capturing interaction among actors, goals and environment in subsystem sets: \( \{S_1, S_2, \ldots, S_n\} \). The diagrams \( G \) and \( I \) can be developed based on either GAIRE system using aggregated item-sets or detailed item-sets on demand. These two diagrams help us to obtain the overall framework of organizational goal and interaction dynamics. Figure 5 shows a sample of Goal diagram, which consists of high-level goals and their subgoals for algorithm plug-in, configuration and execution.

---

**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
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) [19], called POMDP\_AEI. The POMDP\_AEI model is a six-element tuple:

\[
POMDP_{AEI} = \langle S, A, T, R, O, D \rangle,
\]

where \(O(s_t, r_t)\), \(D(a_t, s_{t+1}, d_{t+1})\) is the probability of making observation \(d_{t+1}\) from state \(s_{t+1}\) after having taken action \(a_t\). Figure 6 shows the POMDP\_AEI model. In addition, we can also obtain an interaction dynamic diagram as discussed in Section 4.5 based on GAIRE model. Moreover, environment can be formally modeled with the following grammar.

\[
\text{<environment>} ::= \text{Environment} \text{<type>} \text{<actor>* [resources] [principles] [processes] [forces] [attributes] [invar-properties]}
\]

Fig. 6. 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 [20], Dynamic System Theory [1], logic-based and reactive approaches, and statistical approaches. Additionally, scenarios, sequences, states and activities are often used when presenting system dynamics visually.

In [3], we demonstrated some techniques in analyzing organizational dynamics. Here we take the POMDP\_AEI model as an instance. In F-Trade, the goal of agent AlgoPluginAgent is to fulfill the registration of an algorithm into the system. Corresponding environment states and actions of the agent are listed in Table 3 and 4 (excerpt, complete is available from [3]), respectively. The state-action chain of this agent interacting with its environment can be modeled as shown in Figure 7. 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 \& apr.dependee = PluginInterface \& Fulfilled() \rightarrow [0 \leq t_2 \text{ CheckAlgorithmValidity}] \leq t_1 \text{ AcceptPluginRequest.Fulfilled()})

Table 3. AlgoPluginAgent environment state list

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

Table 4. AlgoConfigureAgent action list

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a_1)</td>
<td>receive register request from PluginInterface</td>
</tr>
<tr>
<td>(a_2)</td>
<td>register and configure ontologies into base C with help from OntologyService (H)</td>
</tr>
</tbody>
</table>

Fig. 7. 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, SODA, etc., 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. OMNI includes formal...
The organizational metaphor is effective in analyzing multi-agent systems and particularly open agent systems. However, some existing problems cannot be handled. To this end, we have demonstrated a new approach for dealing with these problems through the analysis of OCCLUS-like systems. We have also shown that the ORGANISED framework can be used for analyzing and designing agent systems both visually and formally.

7. CONCLUSIONS AND FUTURE WORK

The analysis of OCCLUS-like systems reveals that the proposed framework can be used for analyzing and designing agent systems both visually and formally. The ORGANISED framework provides a clear and structured approach for analyzing and designing agent systems. Future work will focus on extending the framework to handle more complex multi-agent systems and exploring its practical applications.