A Layered Agent Oriented Engineering Approach for Building Complex Systems

SeongKee Lee, SungChan Cho, HyengHo Lee, ChanGon Yoo, JungChan Park, JaeHyun Park, HoSang Yoon, CheolHo Kim
Department of Information Software
Agency for Defense Development
Songpa P.O.132, Songpagu, Seoul 138-600
Republic of Korea

Abstract: This paper shows how to model complex environment through agent oriented world into agent-based system by decomposition to agent, abstraction to agent, layering to agent oriented model and seamless engineering for agent-based system. Based on agent oriented technology, this paper proposes the layered engineering approach for building complex systems. The layers consist of five layers such as infrastructure, agent, organization, coordination, constraint layer. Each layer executes engineering activities using proper diagrams or methods. This paper shows the usefulness of proposed method by applying the proposed method to the design of a battle simulation system. The agent oriented design method is a systematic and efficient engineering method for complex systems.

Key-Words: Complex system, agent oriented model, agent-based system, layered engineering, coordination, system design, simulation system

1 Introduction
Computing environment is changing according to internet provision, computing power increase, software technology improvement. The characteristics of recent complex computing environment are as follows:

1) System of systems : distributed, interoperation, heterogeneity[2]
2) Intelligence : delegation, autonomy, human-machine interface[4]
3) Service oriented : user centered, real time, wireless, ubiquitous[3]

In order to build the system comprised of the features of complex computing environment described above, complex computing environment must be modeled in natural and designed in systematic. The traditional function oriented development methods such as process oriented or object oriented are not sufficient to adopt the features of complex computing environment. Process oriented methods don’t adopt almost three features. Object oriented methods don’t include intelligence feature. More natural systematic modeling and system design method, which adopt the features of complex computing environment, is required. Recently, agent oriented technology has been studied as one of the enabling technologies for modeling and design of complex system[6]. Agent is a software component that has the ability to autonomously undertake action and interacts with other agents to achieve its goal. Using autonomy, reactivity, mobility and goal-directed characteristics, agent or agent-based system achieves its goal on complex computing environment[1]. Based on agent oriented technology, this paper proposes a method which models complex computing environment in the agent oriented view and designs agent-based system. Section 2 explains the general useful concepts applied to model complex system. Section 3, based on the concepts described in Section 2, discusses our approach for agent oriented modeling. Section 4 expands the modeling method for the design of an agent-based system. In order to show the usefulness of the proposed method, Section 5 applies the proposed method to the design of a simulation system.

2 Modeling complex system
Some concepts to model complex system have been used. They divide, simplify, constraint complex problem into more simple, manageable and understandable units. This section addresses the useful concepts to model complex system in the agent oriented view.

1) Decomposition: divides a large system into set of small subsystems. The small subsystems can be easily executed and are manageable. By integrating small subsystems, we can build the large system. In the agent oriented view, we may divide
the complex system into a set of agents.

2) Abstraction: represents complex system into simple system conceptually. It focuses on the interested parts of complex system. Other detail parts are hided for simplicity. Using conceptual models that have only essential features necessary to represent or solve complex problem, we can understand and analyze complex system. In the agent oriented view, complex system may be modeled by agents[18].

3) Layering: separates system functions into layers according to their roles or responsibility. Complex system can be represented a layered system in which each layer plays a role of it. In general, layering concept helps understand and design complex system. This concept may be used to model complex system in the agent oriented view[7].

4) Seamless engineering: builds the modeled complex system into a real system. Systematic procedure, activities and interrelation among activities are required to engineer system. Seamless engineering method is required to design an agent-based system[11,13].

3 Agent oriented modeling

This section explains how to map real complex world into agent oriented world. In the agent oriented view, complex environment is decomposed into domains and each domain has agents. Each agent in domain interacts with other agents to achieve its goal or role. For effective interaction, it is necessary to coordinate agents’ interactions. Fig. 1 shows the agent oriented view of complex computing environment in abstract.

As we described in Section 2, a layered architecture is useful to define complex system. Agent oriented world shown in Fig. 1 can be efficiently refined by some layers in which each layer plays a role of modeling agent oriented world. The layers are as followings: 1) infrastructure layer which defines communication environment among agents, 2) agent layer which defines agents acting on infrastructure layer, 3) organization layer which defines agents’ interaction type and classifies them into domains, 4) coordination layer which mediates the interaction among agents, 5) constraint layer which examines the efficiency of design of agent-based system. In layered model, lower layer provides services for higher layer so that higher layer can execute its role[9].

4 Agent based system design

As we described in Section 2, seamless engineering is useful to build a real system from complex system. We need a seamless engineering approach to build agent-based system in systematic and efficient. The layered agent oriented model shown in Fig.2 can be used to engineer an agent-based system for complex system. This section discusses an engineering approach of agent-based system by refining each layer in Fig.2 and interconnecting the activities or the products of each layer.

4.1 Infrastructure Layer

This layer provides the operation environment that agents can act on. The primary design factors are communication protocol and network environment. For example, TCP/IP on internet, LAN protocols on intranet. If agent accesses data sources on internet, then infrastructure must be defined as TCP/IP.
We can classify domains using relation graph explained in Subsection 4.2, domains independently interact with each other. By domains which some agents belong to. These agents would be separately placed in different place. Otherwise, two agents would be placed in same place(domain). In result, agent-based system is organized by the structure, we must decide the place where agents be placed. In tree or group structure(P), each agent directly communicates with other agents under its local facilitator. In tree mediate the interaction among the agents connected directly with it. The superior agent in root controls the interaction among middle superior agents. Other structure(G) is to group agents with local facilitators that store the information necessary to control the interaction among agents under that. Each agent is fully connected with peer agents under its local facilitator. In tree or group structure, we must decide the place where agents belong to. The interaction between agents can be classified into local or remote. If interaction between two agents is local, then two agents would be placed in same place(domain). Otherwise, two agents would be separately placed in different place. In result, agent-based system is organized by the domains which some agents belong to. These domains independently interact with each other. By using relation graph explained in Subsection 4.2, we can classify domains.

### 4.2 Agent Layer

This layer identifies the agents and defines their features, in particular, its behaviors. Each agent has its role, behaviors and knowledge. At first, agents can be extracted by analyzing application description. We extract goal, sub-goals and tasks and represent them in hierarchical structure using goal hierarchy diagram. After grouping similar tasks in a diagram, we assign role to each group. Then we find actor(agent) that can execute the identified role. In order to define agents’ behaviors, it is useful to examine the relationship among agents using graph-like form. Node in graph represents agent and link represents interaction between agents. Message in link represents relations between agents. Since each agent has responsibilities to process incoming message and to reply to that, the incoming messages become the behaviors of agents receiving message.

### 4.3 Organization Layer

This layer is defined using the agents of agent layer. The organization of agent-based system is diverse according to the role or the interrelationship among agents. The primary structures that organize agents are classified as follows[8]. In Peer-to-Peer structure(P), each agent directly communicates with other agents without intervention. Agents are fully connected without central control. Tree structure(T) has a hierarchical structure in which superior agent (facilitator) is parent and the agent under superior agent becomes child. Superior agents in middle of tree mediate the interaction among the agents connected directly with it. The superior agent in root controls the interaction among middle superior agents. Other structure(G) is to group agents with local facilitators that store the information necessary to control the interaction among agents under that. Each agent is fully connected with peer agents under its local facilitator. In tree or group structure, we must decide the place where agents belong to. The interaction between agents can be classified into local or remote. If interaction between two agents is local, then two agents would be placed in same place(domain). Otherwise, two agents would be separately placed in different place. In result, agent-based system is organized by the domains which some agents belong to. These domains independently interact with each other. By using relation graph explained in Subsection 4.2, we can classify domains.

### 4.4 Coordination Layer

When system responses to service request, system must control which domain is assigned to reply and which agent is appropriate to answer to the request. In order to control this situation, coordination method is needed. There are some coordination methods such as matchmaker(M), broker(B) and contract-net(C)[16]. Matchmaker and broker method mediate between requester and server. In case of matchmaker, after matchmaker matching requester and server first, they directly communicate with each other. Broker continues to intervene between them after match. They indirectly communicate with each other via broker. Contract-net method directly interacts with between requester and server. These methods are used to mediate both the interaction among domains and among agents. In view of communication primitive, when matchmaker or broker method is used, 2n+4(n is the number of agents or domains) interactions between requester and server are required. In case of contract-net, 3n+1 are required.

### 4.5 Constraint Layer

This layer verifies that system is designed in satisfactory. There are two constraints necessary to build more efficient system. One is goal achievement. This checks that system satisfies user requests. The other is system efficiency. This checks how efficient the system is designed. This paper considers only system efficiency. The useful measurements to analyze agent-based system efficiency are the load balance of domains and agents and the selection load to find next domain and agent[5]. They can be analyzed using proper metrics. In particular, the selection load to find next domain or agent is an important measurement factor that can affect dynamic characteristics of agent-based system. To analyze this factor, we can define architecture models of agent-based system by combining the organization structures described in Subsection 4.3 with the coordination methods described in Subsection 4.4. For example, architecture models are as follows: P-C(Peer-to-peer structure, Contract-net for agents), P-M(Peer-to-peer structure, Matchmaker for agents), P-B(Peer-to-peer structure, Broker for agents), G-C-B(Groups structure, Contract-net for facilitator, Broker for agents), G-C-M(Groups structure, Contract-net for facilitator, Matchmaker for agents), T-B-B(Tree structure, Broker for facilitator, Broker for agents), and so on. About 16 architecture models can be extracted. Based on the communication primitive required for interaction between requester and server, we can measure the selection load to find next domain and agent for...
each model. P-M and P-B in peer-to-peer structure are efficient. G-C-M and G-C-B in group structure are efficient. T-C-B and T-B-B in tree structure is efficient model[10].

5 Application to modeling and simulation system design

This section applies the engineering approach for agent-based system described above to the design of a battle simulation system. In general, battlefield is complex environment including combat entities such as combatant, unit and weapon, natural entities such as terrain and tree, artifact entities such as obstacle, weather, and so on. During engagement, they have to interact with each other. We engineer an agent-based system for a virtual simulation system for small sized unit’s tactical exercise in synthetic battlefield. Simplified application description for battle simulation system is as follows.

This system is a battle simulation system networked virtual environment that enables the exercise participants to share with common 3D synthetic virtual battle-space and execute interactive combat on it. The exercise participants include commander, combatant, unit, weapon system and etc. All trainees are connected with on local area network environment. This system provides the functions necessary to unit’s tactical exercise preparation, execution and analysis. Also, in order to simulate various tactics, this system includes semi-autonomous computer generated force(CGF).

5.1 Infrastructure Layer

This layer defines the operation environment that agents of battle simulation system act on and communicate with each other. As described in application description above, the battle simulation system is operated on local area network environment and uses TCP/IP protocol.

5.2 Agent Layer

Analyzing the description of battle simulation system, we can breakdown it into sub-goals and tasks using goal hierarchical diagram as shown in Fig. 3. In Fig. 3, we choose major agents(colored parts) from tasks. They are simulation engine, rendering, commander, unit, weapon, computer generated force and external connection. In the agent oriented view, simulation engine, rendering and external connection are agents without autonomy. Commander, unit and weapon are agents with human decision. Computer generated force is semi-autonomous agent. They interact with each other to execute their roles. Fig. 4 shows the relation among them in graph form. Examining communication messages between agents, we can extract the behaviors for each agent to execute. This paper doesn’t discuss that.

5.3 Organization Layer

After assigning interaction type, local(l) or remote(r), to the relation among agents, grouping the local type agents and separating them from the remote type agents, we can extract the domains which interrelated agents reside in. In our example, since simulation engine and rendering agents are common to all trainees, interaction type is local. Fig. 4 marks only remote type. The interaction type of the unmarked links is local. From the result of grouping agents, five domains are identified as follows: domain1(external connection), domain2(commander, simulation engine, rendering), domain3(unit, simulation engine, rendering), domain4(weapon simulator, simulation engine, rendering), domain5(CGF, simulation engine, rendering). Fig. 5 shows the result of domain identification. The interaction type among domains is basically remote. Through these procedures, we can organize an agent-based battle simulation system.

![Fig. 3 Battle simulation goal and tasks](attachment:image.png)
5.4 Coordination Layer

It is necessary to control the interaction among agents or domains. The primary coordination methods are matchmaker, broker and contract-net. As shown in Fig. 5, the organization of agent-based battle simulation system is tree structure in which external connection agent is root. Agents in each domain can interact with agents in other domains via external connection agent of domain1. External connection agent plays a role of facilitator that mediates the interaction among domains. In order to interconnect with domains, we consider RTI (runtime infrastructure). RTI is a CORBA-like middleware software that allows simulation systems to interoperate with each other. In system organization of Fig. 5, RTI recognizes a domain into a federate and supports the interaction of domains as it acts like a broker. In result, coordination method of this battle simulation system is broker[7].

5.5 Constraint Layer

Based on the result of previous layers, the architecture model of this agent-based battle system is T-B (tree structure, broker). If agents and domains increases, then other external connection facilitators would be added and architectural model become T-B-B (tree structure, broker for facilitators, broker for agents). This model is more efficient than other architectural models such as T-C-C or T-B-C.

5.6 Implementation

Based on the result of agent-based battle simulation system design, we implement a prototype as shown in Fig. 6.

Each trainee communicates with each other via RTI Exec. RTI Exec corresponds to the domain1 in Fig. 5. It mediates or facilitates interactions with unit, weapon, CGF and commander. The trainees engage with each other in interactive in real time. In particular, CGF semi-autonomously participate in engagement using its combat rules without human intervention.
6 Conclusion

Based on agent oriented technology, this paper proposes a method which models complex computing environment and designs agent-based system. We define a layered engineering approach consisting of five layers such as infrastructure, agent, organization, coordination and constraint layer. Each layer executes engineering activities using proper diagrams or methods. Agent layer uses goal hierarchical diagram to identify agent and shows the relationship among agents in graph form. Organization layer classifies domains by interaction type between agents. Coordination layer decides coordination method according to interaction pattern among agents. Constraint layer defines the architecture models that combine organization structure with coordination method and analyzes the selection load to next agent using communication primitive occurring between agents. If necessary, engineer can act engineering activities again for more efficient system design. In future, more detail engineering method for agent-based system development is required.

This paper shows how to model complex environment through agent oriented world into agent-based system by decomposition to agent, abstraction to agent, layering to agent oriented model and seamless engineering for agent-based system. Since agent technology can be applied to build the diverse system from very small device to large systems to systems integration, the technology usage will be increased. The systematic and efficient engineering method will be essential to engineers. We expect that the approach proposed in this paper would be referenced to establish the method.

References: