Implementation of Role-based Command Hierarchy Model for Actor Cooperation

Jungyoon Kim, Hee-Soo Kim, Jihyun Jang
REALTIMEVISUAL
Seoul, Republic of Korea
jkim@realtimevisual.com, hskim@realtimevisual.com, jhjang@realtimevisual.com

Sangjin Lee, Samjoon Park
Agency for Defense Development
Daejeon, Republic of Korea
sangjinlee@add.re.kr, samjoon@add.re.kr

ABSTRACT

Many approaches to agent collaboration have been introduced in military war-games, and those approaches address methods for actor- (agent-) collaboration within a team to achieve given goals, where the team's abstract mission is translated into concrete tasks for each actor. To meet fast-changing battlefield situations, an actor must be 1) loosely coupled with their tasks and be 2) able to take over the role of other actors if necessary to reflect role handovers occurring in real combat. Achieving these requirements allows the transfer of tasks assigned one actor to another actor in circumstances when that actor cannot execute its assigned role, such as when destroyed in action. Tight coupling between an actor and its tasks can prevent role handover in fast-changing situations. Unfortunately, existing approaches and war-game software strictly assign tasks to actors during design, therefore they prevent the loose coupling needed for successful role handover. To overcome these shortcomings, we have defined Role-based Command Hierarchy (ROCH) model that dynamically assigns roles to actors based on their situation at runtime. In the model we devise "Role" to separate actors from their tasks. Described in this paper, we implement the ROCH model as a component that uses a publish-subscribe pattern to handle the link between an actor and the roles of its subordinates (other actors in the team). Therefore, an actor can indirectly send a message (order or report) to another actor without knowing which actor is recipient. The sender actor is only required to know the relevant roles. The model has been implemented and tested in a military project, and we briefly show the outcomes in this paper.

ABOUT THE AUTHORS

Jungyoon Kim is a research engineer in REALTIEMVISUAL Inc., Seoul, Korea. He has been working on development of military logistics systems and joined the M&S area in 2010. He obtained BS in Aeronautical Engineering from Korea Air Force Academy, received MS. from Texas Tech University and Ph.D from Korea Advanced Institute of Science & Technology (KAIST) in computer science.

Hee-Soo Kim is presently a Ph.D. candidate in Graduate School of Information and Communication at Ajou University and works as a research engineer in REALTIMEVISUAL Inc. He has received M.Sc. in Graduate School of Information and Communication from Ajou University in 2005. Areas of his interest include modeling and simulation, software engineering, and large-scale system infrastructure based on knowledge engineering.

Sangjin LEE is a Senior Researcher in the Modeling and Simulation Division of the Agency for Defense Development in the Republic of Korea. He received a Ph. D in Industrial Engineering from the Korea Advanced Institute of Science and Technology (KAIST) in 2008. He has worked in developing applications of modeling and simulation since 2011. He is interested in composable simulation framework architectures and computer generated forces.

Samjoon PARK is a Principal Researcher and Director of the Modeling and Simulation Division of the Agency for Defense Development in the Republic of Korea. He received a Ph.D. in Industrial Engineering from the Korea Advanced Institute of Science and Technology (KAIST) in 2006. He has worked in optimization for integrated logistics support and developing information systems since 1989. He is interested in composable simulation framework architectures, computer generated forces, and munitions effectiveness.

Jihyun Jang has been working as an engineer in REALTIMEVISUAL Inc. for two years. He has received B.S. and MS. in Bio & Brain Engineering from Korea Advanced Institute of Science and Technology (KAIST).

Implementation of Role-based Command Hierarchy Model for Actor Cooperation

Jungyoon Kim, Hee-Soo Kim, Jihyun Jang
REALTIMEVISUAL
Seoul, Republic of Korea
jkim@realtimevisual.com, hskim@realtimevisual.com, **jhjang@realtimevisual.com**

Sangjin Lee, Samjoon Park
Agency for Defense Development
Daejeon, Republic of Korea
sangjinlee@add.re.kr, samjoon@add.re.kr

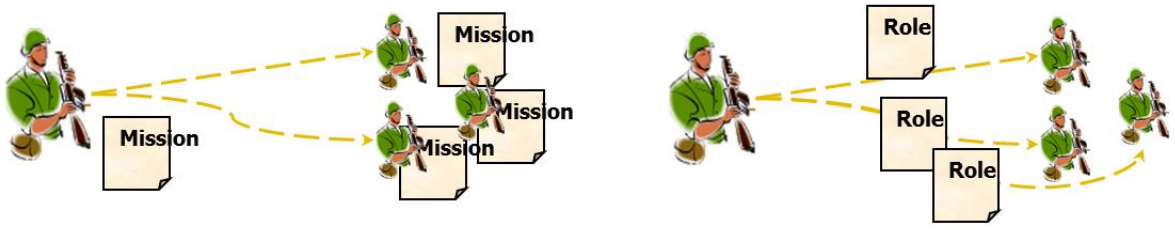
INTRODUCTION

War games model and simulate military elements (combatants, activities), test the effectiveness of combat participants and their tactics or doctrines in an anticipated real engagement with a reduced budget. In a war game, *Actor* is a structural element, and *Tactics* is a behavioral element. Table 1 provides two important definitions to these terms.

Table 1. Element Naming in Existing Models

Terms	Meaning
Actor	− The agent in agent based simulation area. (We use the term 'agent' differently in this paper.) − Individual entities (rifleman, tank, aircraft …) that interact with physical elements − Group of entities (platoon, company, battalion …) that exist within a virtual unit
Tactics	− Actor's behavior. Realized into simulation scenario. Simulation engine translates the scenario into Actor's tasks and the transition requires ways to interpret group's Tactics into Actors' tasks (cooperation, specifically mission transfer). − *Plan* (a realized Tactics) defines participants and their goals in a simulation. The participants (combatants) should achieve goals collaboratively.

Existing models are successful in modeling the activities of aggregate units or individual entities. Some follow modularized model development, making them reusable and composable (Logsdon, Nash, & Barnes, 2008; Ternion Corporation, 2012; VT MÄK, 2012). However, the Actors and Tactics are tightly coupled, making some situations difficult to represent within a simulation. Since a mission is just a sequence of tasks given and attached to an Actor, without the ability to transfer an Actor's tasks to another, the inability of an Actor to continue its mission within a simulation necessarily results in the loss of its mission within the simulation. This is an inappropriate representation of real-world situations in which a mission-transfer to another live combatant occurs frequently as a combatant injury or other casualty. The left side of Figure 1 illustrates this situation.



- **Figure 1. Tightly coupled missions and separated missions (Roles)**

To overcome such shortcoming, we have introduced Role-based Command Hierarchy (ROCH) (Kim & Lee, 2013) based on the concept of right side of Figure 1. It separates the Tactics from Actor components using a role concept. *Role* is a logical connection between an Actor's Tactics and its subordinates (other Actors) and it is defined at design time. The Actor is dynamically bound to its subordinates through Roles according to situations at runtime. This architecture enables simulations to be more composable, reusable, and scalable, in that a Role can be reused with minimal modification of its Actor and an Actor can be reused without considering its mission. After some minor

corrections ROCH has been implemented as a part of Korean military simulation. This paper is about explanation of ROCH and of actual implemented interfaces.

REATED WORKS

US Army has developed a simulation product line, OneSAF, to supports various military simulation areas including engagement analysis, personnel training, and acquisition decision (Logsdon, Nash, & Barnes, 2008). It enables military users to cover a wide range of constructive and virtual simulator solutions. Additionally, various commercial simulation architectures are successful in modeling tactics of aggregate or individual entities. Among them, FLAMES and VR-Forces model physical domains (land, sea, air, and space) at a wide range of fidelity and resolution levels (Ternion Corporation, 2012; VT MÄK, 2012). They are founded on the concept of component based development. The elements of a simulation such as units, entities, and their tactics are implemented as modularized components.

OneSAF, FLAMES, and VR-Forces are highly reusable. Their component based modularity helps developers to rapid develop M&S systems with low cost. However, they are limited to model tactics in real combat and do not effectively reflect fast-changing combatants' missions. (e.g., a combatant may be removed from its mission such as killed in action, with the mission left usually transferred to another combatant.) Reflecting such situation requires a more complex specification at design time to describe all possible task handover situations.

Other related works have similar problems, such as hierarchical agent control (HAC) (Atkin, Westbrook, & Cohen, 2001), commander model (CB) (Vakas, Prince, Blacksten, & Burdick, 2001), command-based multi-agent system (CMAS) (Song, & Yang, 2006), tactical team behavior (TTB) (Bisht, Malhotra, & Taneja, 2007), agent-group-role (AGR) (Ferber, Gutknecht, & Michel, 2004), and so forth. Each model is helpful to represent the tactical behaviors of aggregate units and to enable the units to perform the tactical activities. However, the activities are tightly coupled, making it difficult to reuse units or activities.

Table 2. Element Naming in Existing Models

Modeling Elements	OneSAF	FLAMES	VR-Forces	CSFA
Actor	Actor	Unit	Entity	Unit
Atomic actor	Entity	Unit	Entity	Entity
Aggregate actor	Unit	-	Aggregate Entity	Aggregate Unit
Physical model	Physical Agent / Physical Model	Equipment	Sensor / Actuator	Equipment Agent/ Equipment Model
Behavior logic	behavior Agent / behavior Model	Cognition Model	Controller	behavior Agent
Intra unit communication	Trigger	Specific interface	Port	Trigger
Inter unit communication	Event	Message	Message	Event
Cooperation	behavior Agent	Cognition Model	Controller	Behavior Agent

Common Model

We surveyed four simulation architectures to determine their representations of their elements. They are three outstanding architectures, OneSAF, FLAMES, and VR-Forces, which can be applied to military or commercial areas, and one architecture, Composable Software Architecture Framework (CSFA) (Petty, Kim, & Byun, 2014), developed by us as a national defense research project to show the feasibility of composable and modular simulation. CSFA can be regarded as an outcome from benchmark of OneSAF, reflecting indigenous features for Korean Army.

Since OneSAF aims to reflect global level military operations including non-conventional wars such as anti-terrorist activities, its vast list of tasks (behaviors) is inappropriate to the Korean situation.

As a result of the survey, we have introduced a *Common Model* by comparing the elements of the architectures above and categorizing them with their similarities into modeling elements as shown in Table 2. Thus Common Model can be regarded as an average categorization of existing models and their elements. Because the architectures are component-based warfare M&S software products, in many aspects most elements in a same category are mainly similar concepts or the same character with just different names.

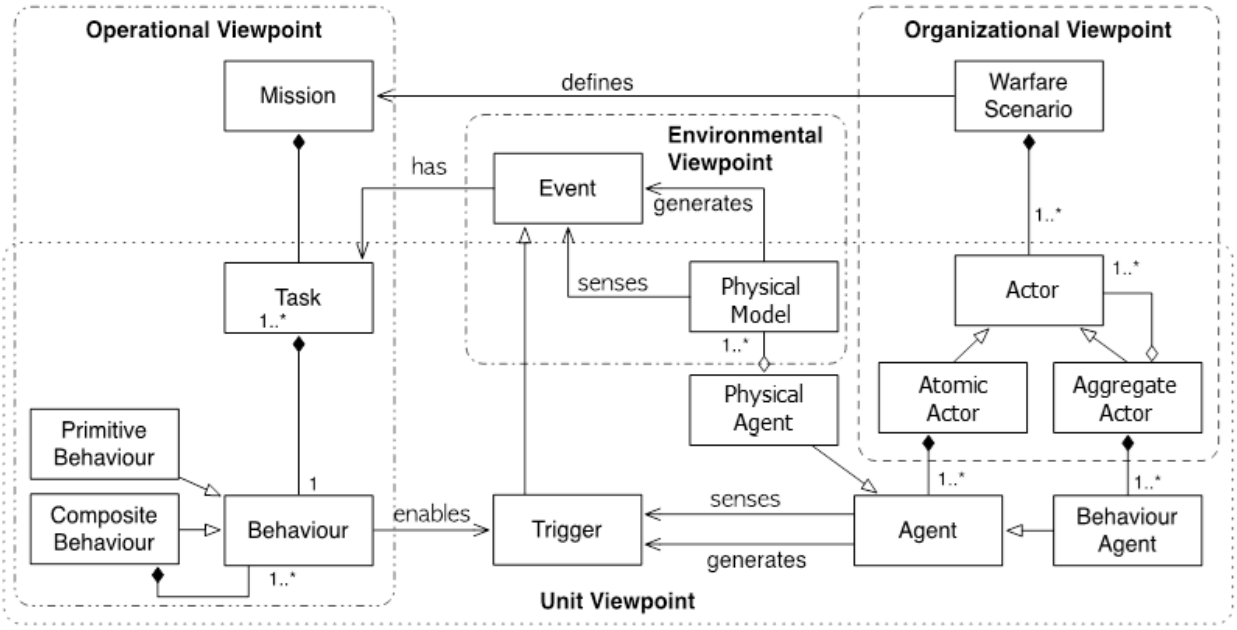


Figure 2. Common Model from existing models

Elements in some software architectures can be seen as structural or behavioral. Figure 2 shows such elements of Common Model introduced from the survey. Most names, such as Actor and Behavior, mainly borrowed from OneSAF. In the model, a *Warfare Scenario* includes organizational information for each blue or red force in that an *Aggregate Actor* is organized by Atomic Actors or other types of Aggregate Actors. A Warfare Scenario also defines the *Mission*, which is composed of *Tasks*. An Actor can be Atomic or Aggregate: *Atomic Actor* (an entity in a battlefield) has no subordinates and is composed of Physical and Behavior Agents, while an Aggregate Actor (or organizational unit) includes group of Actors and is composed only of Behavior Agents. Relations between a higher Actor and its subordinate Actors reflect a command chain. An Actor also includes Agents that are equivalent to the interfaces of a component and are categorized into physical and behavioral: A *Physical Agent* represents one of Actor's parts such as sensor, mobility, weapons. A *Behavior Agent* decides the Behavior for its unit. Inside an Actor, Agents communicate with other Agents through Triggers. Actor detects or makes changes in the environment through its Physical Models/Agents.

The behavioral elements of Common Model are Behavior, Task, Mission, Event and Trigger. A Behavior can be Composite or Primitive Behavior. A *Composite Behavior* is composed of other behaviors that are arranged in order. A *Primitive Behavior* controls its relevant Behavior Agent by generating Triggers to initiate Actor's Behavior. Tasks are delivered to an Actor in the form of an *Event*. A *Task* is specified as a combination of Behaviors. Actors perform Tasks by executing its Behaviors. A *Mission* is a conceptual goal that an Actor must achieve and is composed of a sequential set of Tasks. A Mission forms the long-term plan that Actor executes. Finally, *Trigger* is the transformed Event capture from the outside of an Actor, and initiates Actor's Behavior. Trigger is equivalent to Event, for example "being attacked" as a change of the environment is delivered in the form of an Event and then is called Trigger once it gets inside of an Actor. At the beginning of a simulation, each Agent subscribes to some Events in which the Agent is interested, then once an Event happens simulation engine notifies the Agent that subscribed for the Event. Finally the Agent acts according to the Event (publish-subscribe pattern).

Common Model can be seen with four different viewpoints as circled with dotted lines in Figure 2: operational, organizational, environmental, and unit viewpoints. The Operational Viewpoint is the aspect of tactics executed in a real battlefield. It implies the behavioral elements of a simulation. The Organizational Viewpoint is the aspect of combatants participating in an engagement and implies the structural elements of a simulation. Environmental Viewpoint is simply the aspect of interaction between environment and Actor. The Event is transferred to Actor through Physical Model. The *Physical Model* represents any mediator which can detect any change in environment or can give effect to environment, such as sensors, fire arms, mobility, or communication devices. Unit viewpoint is the aspect of an Actor composition. It sees an Actor's composition and executes its mission.

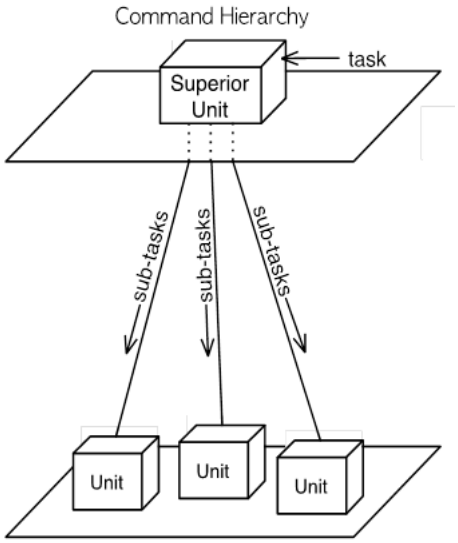


Figure 3. Command Hierarchy in Common Model

Limitation of Existing Models

One limitation of the Common Model is that it is difficult to reflect certain situations (such as role-transfer aforementioned). As an Actor and its Tasks are tangled each other through Warfare Scenario and Mission, it is hard to reuse an Actor or Task alone. As shown in Figure 2, each Actor has its own missions, and the schema can be redrawn as Figure 3. In Figure 3, Superior Unit receives task and each Units receives subtask from Superior Unit. Generally such task and subtasks are defined at design time (scenario editing), considering that each subtasks are to achieve Superior Unit's task. As a result, the task and subtasks are tightly coupled with its Actors (the units).

In the scheme of Common Model, task cannot be swapped with other units' tasks, thus if the owner Actor of the task is destroyed and removed from the simulation, its task also will be removed. HAC (Atkin, Westbrook, & Cohen, 2001), CM (Vakas, Prince, Blacksten, & Burdick, 2001), and TTB (Bisht, Malhotra, & Taneja, 2007) focus on modeling of tactics with little consideration of unit components' composability. CMAS (Song, & Yang, 2006) and AGR (Ferber, Gutknecht, & Michel, 2004), can specify the tactics of units as interaction between roles that they play but do not clearly provide the method to separate units from the roles. This tight coupling means that a superior unit should be dependent on its subordinates or vice versa.

ROLE BASED COMMAND HIERARCHY (ROCH)

Figure 4 briefly illustrates the basic concept of ROCH architecture. The Goal of ROCH is to separate tactics from units in order to overcome the drawback of tight coupling. The distinguished part of ROCH is Role. It is a virtual seat that indicates an actual subordinate unit, which is not predetermined at design time. It bounds Tactics (set of Tasks) to an Actor at runtime, making the subordinates (Unit) of the Superior Unit tactically behave. A sub-task is the subordinate's contribution to achieve the goal of Superior Unit's task. A Role has a set of sub-tasks that its owner (Unit) should be able to execute. Thus the set of all Subtasks can be regarded as the Tactics for the Superior Unit.

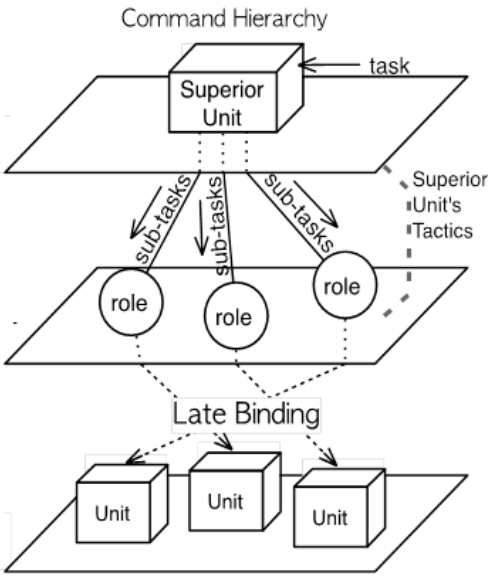


Figure 4. ROCH Architecture

ROCH is composed of two artifacts: a Meta-Model and a Framework. The *Meta-Model* is used to specify Actors and Plans as a machine-readable representation using XML, which is different from the previous role-based approaches (Xu, Zhang, & Patel, 2007; Cabri, Leonardi, & Zambonelli, 2003; Becht, Gurzki, Klarmann, & Muscholl, 1999; Hahn, Madrigal-Mora, & Fischer,

2009) where role (tactics) is usually hard coded within its player at design time. The Meta-Model allows the tactics of Unit to be freely modified by users without knowing Unit's Condition. The *Framework* provides the facilities for each Unit to execute their Plans with mechanism dynamically assigning Roles considering situation at runtime.

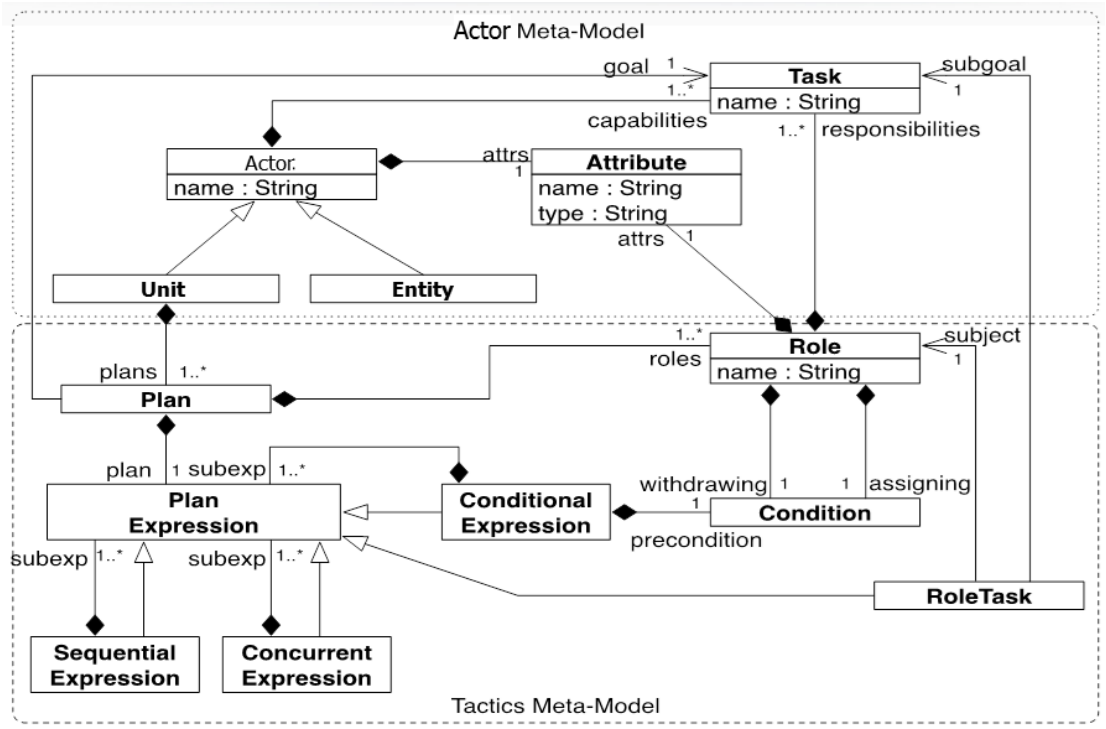


Figure 5. ROCH Meta Model

Figure 5 shows Meta-Model in UML. Meta-Model is based on the context of the model-driven architecture (Object Management Group, 2012) and the Common Model mentioned in previous chapter. Meta-Model is composed of Actor Meta-Model and Tactics Meta-Model. In Actor Meta-Model *Attribute* is a variable which comprises the Actor's state and *Tasks* is what Actor performs. Tactics Meta-Model represents the plan in which Actor and its subordinate Actors are specified. A *Plan* is composed of Roles and a Plan Expression. Role is the distinguishing characteristic of ROCH and has attributes and tasks (responsibilities) and two Conditions (assigning and withdrawing Conditions). Role can be regarded as the requirements to the Actor. For a subordinate Actor (the Unit in Figure 4) to take the role, it must have the attributes and capability to perform the tasks of the role.

Meta-Model defines Role conditions which are capabilities for a superior Actor that dynamically selects a player of the Role among their subordinate Actors according to their capabilities. The *Assigning Condition* requires that a subordinate can play the Role assigned. The *Withdrawing Condition* indicates that a subordinate, which is playing an assigned Role, cannot play it anymore. In implementation, we removed the Withdrawing Condition from the architecture since it is equivalent to not meeting an assigned condition. However, this is an important concept in ROCH, so we kept the Withdrawing Condition in the ROCH Meta-Model. *Plan* is the series of the sub-tasks to be achieved by the superior Actor. It is represented to be a sequential, concurrent, conditional, or *Role Task* expression (other expression) for achieving a task assigned by a simulation user or its superior Actor. A role-task is the terminal expression to represent a task that a role's player should perform.

Framework is another major part of ROCH, as shown in Figure 6 and 7. The main function of the Framework is to execute plans under the behavior agent of a Unit in Figure 6. The state informing function is for units to inform its superior Actor about its own capabilities and attributes. It is called when the Actor participates in its superior's group or any capability/attribute of the Actor is changed. In the publish-subscribe pattern for the handling link between an Actor and Roles of its subordinates (other Actors), an Actor can indirectly send message (order or report) to another Actor without knowing which Actor is recipient. The sender Actor is only required to know the relevant Roles. *Plan Lib* loads the Plans for a Unit. Also, it selects and activates the Plan for the Task received from a

simulation user or a superior unit. The *Role Manager* assigns/withdraws the Roles according to assigning/withdrawing Conditions, and the *State Manager* maintains the states of its owner (Unit) and the subordinates. The Framework operates based on *Scenario* that includes the organizational information (hierarchy). Scenario also has references to the configuration of a Unit (subordinate). Unit includes the unit model, an implementation package. Its state informing function is to report about its own capabilities and attributes, and Role assigning process.

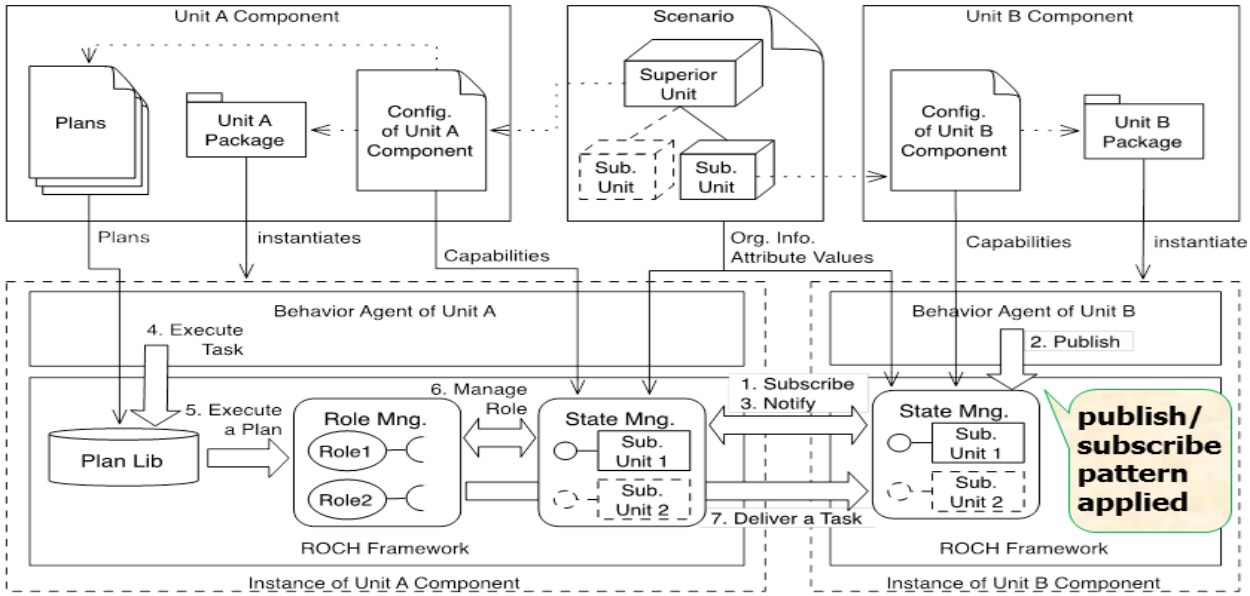


Figure 6. ROCH Framework

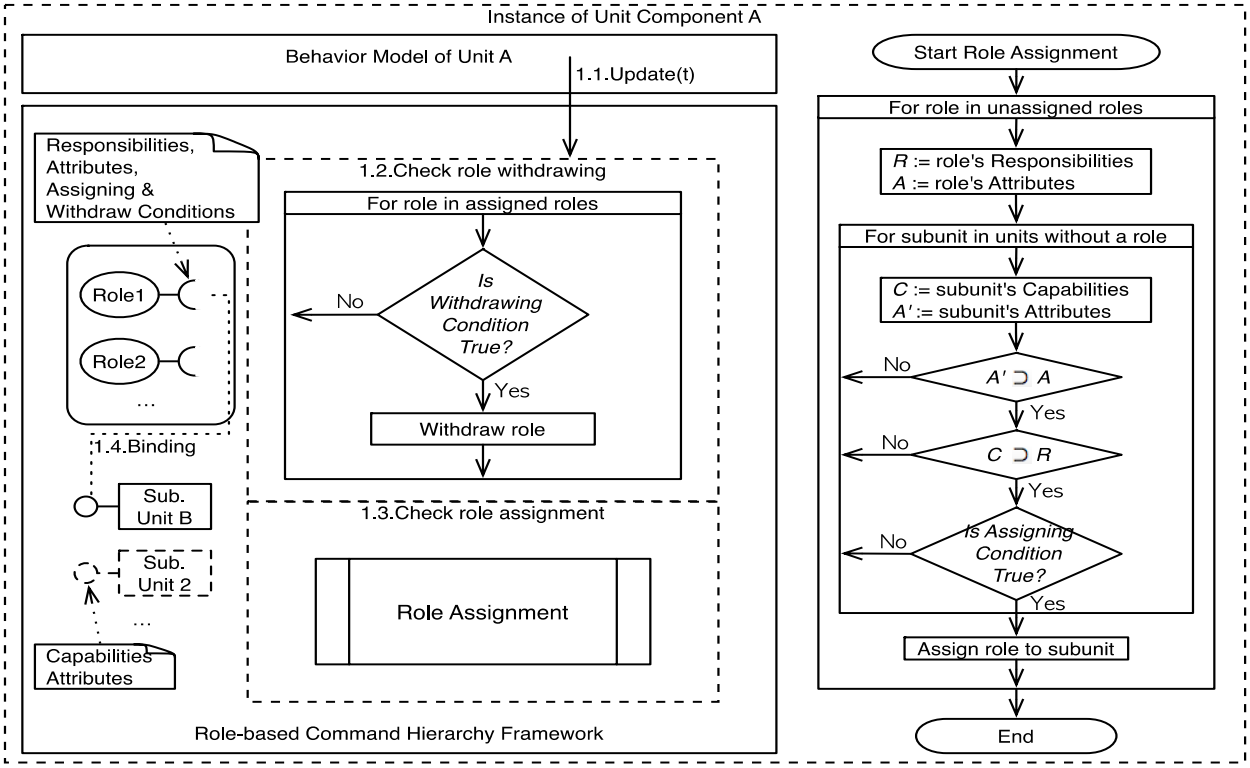


Figure 7. Role Assignment and Withdrawing

The plan execution function is to execute the plan. When a Unit receives a task (4), its Plan Lib selects and activates the Plan (5). Role Manager assigns the proper subordinates to the Roles specified in the Plan (6). Each Role is bound to a subordinate, and a series of sub-tasks are delivered to the role-playing subordinate (7). If a role-playing subordinate cannot play the Role (i.e., the subordinate's state meets the withdrawing condition), the Role is reassigned to another possible subordinates by Role Manager.

In Figure 7, the Behavior Agent of a Unit periodically calls the *Update* function (1.1). It checks if there are Roles to be withdrawn from/assigned to its subordinates using Roles' withdrawing/assigning Conditions (1.2 and 1.3). These steps use the information of Roles and subordinates obtained by State Managers. The last step is to assign a Role to a subordinate if the assigning condition of the Role is met by the current state. After Role assignment, a pair of Role and subordinate is created in the framework (1.4).

IMPLEMENTATION

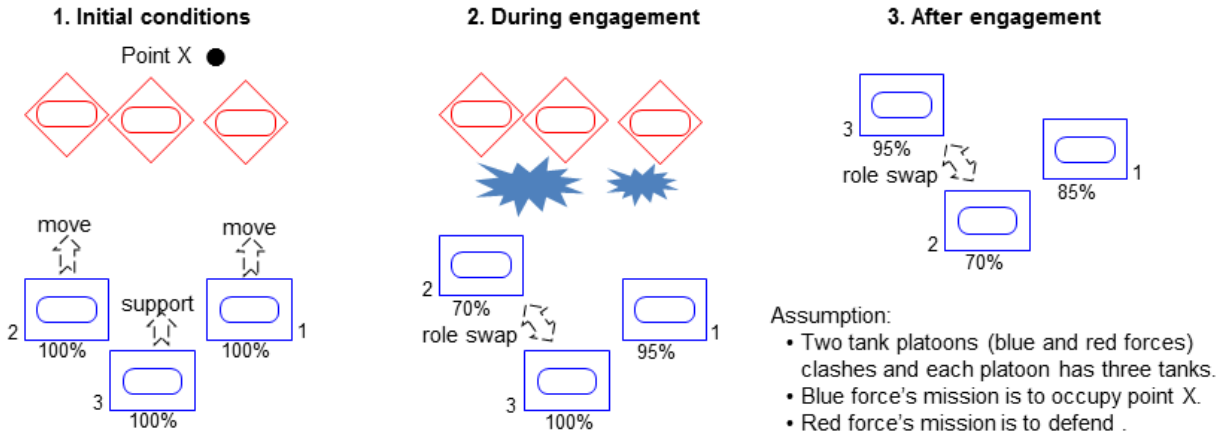


Figure 8. Simple Scenario for Test and Expected Result

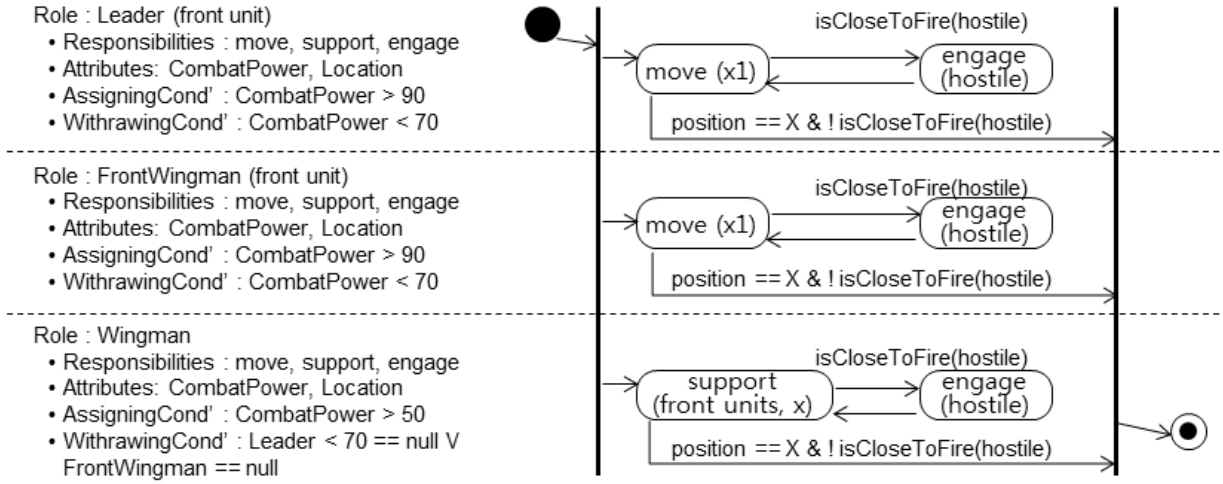


Figure 9. Role Definition in the Scenario

To show how ROCH realizes the mission transfer situation in a war game simulation, simple combat scenario is applied as shown in Figure 8 and 9. The scenario defines that the Blue tank platoon engages with a Red tank platoon. For Blue, a platoon model and three tank models are defined. The plan for the platoon is to occupy the location Point X. For Red, another platoon is defined with three tank models. Its plan is to defend against the Blue tank platoon. Figure 9 shows the Role definitions for each tank in Blue side. Roles are Leader, FrontWingman, and Wingman. Leader is to lead the platoon at the front row in formation, FrontWingman keeps position to be in the same row with Leader (at front), and Wingman is to support for the front row tanks. Leader and FrontWingman

must have their CombatPower over 90 while Wingman can have over 50 (The value of Combat Power can be regarded as percentage). We expected the Wingman takes over the role of Leader or FrontWingman.

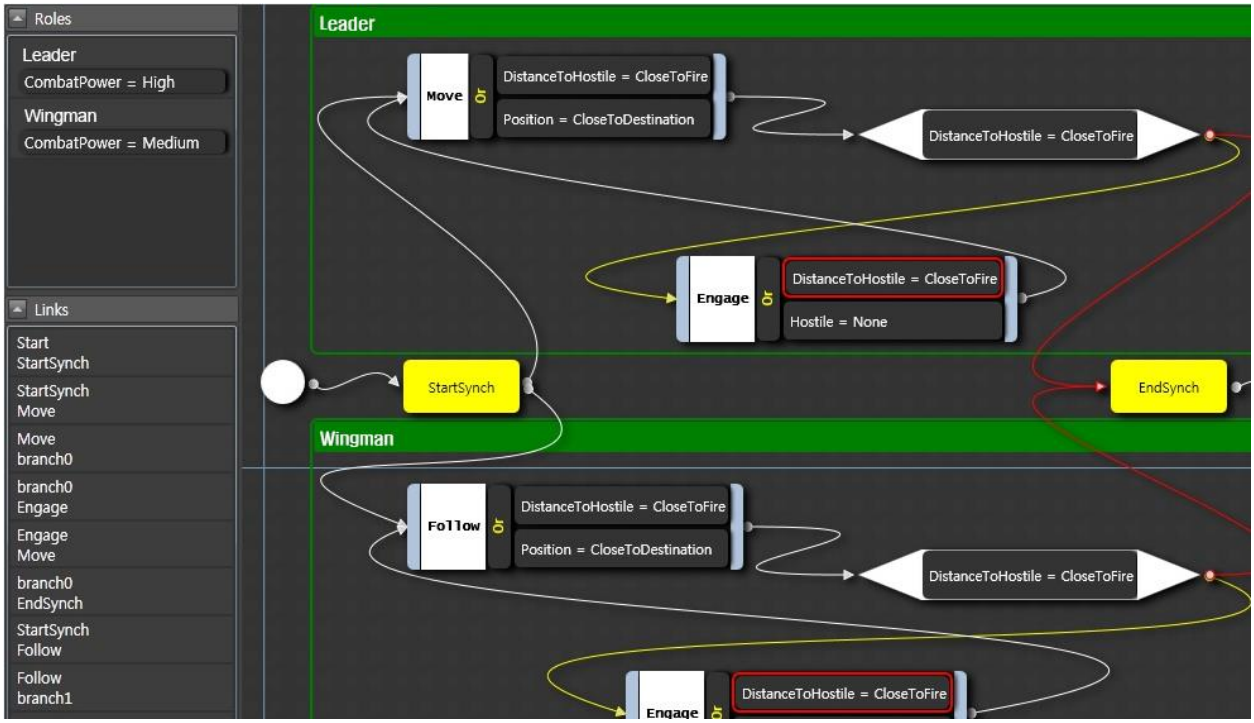


Figure10. Screen Capture for Behavior Editor

Even though it is simple scenario; it contains the context of the benefits of ROCH. Figure 10 is the captured UI screen of actual simulation software that incorporates ROCH as a part of its behavior editor. In the screen, the green boxes are Roles and they include behaviors that are connected to their next behaviors. The Leader executes "Move" behavior then engages with the encountered enemy. Wingman executes "Follow" behavior as reserve force for support. The left upper pane "Role" shows the two Roles defined in the Behavior Editor. It says if Leader loses power under "High" and there is any Wingman that has power over "High" then the Leader and the Wingman swap their roles. (In this case, the Leader does not mean actual leader in real combat. It rather means the tank who leads at the foremost in the formation.) The threshold value for High and Medium are defined in different UI. In implementation, two Role assignment conditions (assigning and withdrawing) became one, as aforementioned. Only the assigning condition is needed for the two

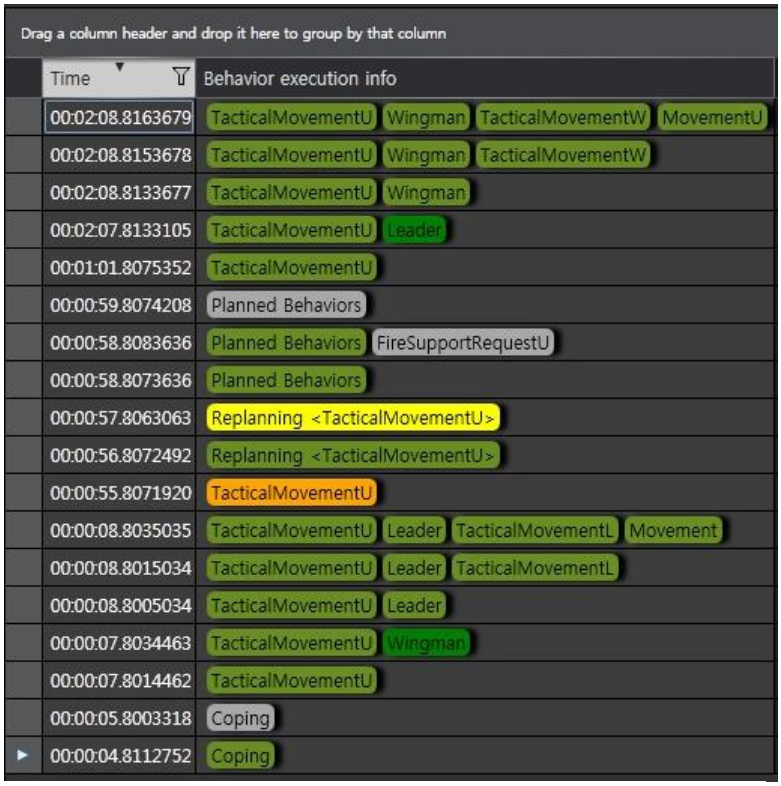


Figure 11. Part of Simulation Log for #3 Tank (Time flows from bottom to up)

conditions. As shown in the Roles pane the condition does not differentiate assigning or withdrawing. Through the implementation of ROCH, we can have confidence on our claim that composability and reusability are improved. As Role links superior and subordinate as medium between them (as no direct dependency between platoon and tank components), a component of tank with predefined capabilities could be reused with little consideration about its tactics.

Figure 11 shows the part of the simulation log for Tank #3, defined in figure 8. The log table shows the action record of the tank, with each row giving the timestamp of behavior occurrence. Time flows from the bottom to upper. Each row begins with a behavior which is the first rounded box. Whenever the tank checks whether the behavior is proper for a certain situation at a time, the rounded box appears. Also the tank checks role condition then the second rounded box appears. If the role in a row (usually the second box in the row) is changed by the simulation engine, then a rounded box with different name is shown in the later (upper) row. In the figure the tank was assigned as Wingman at the beginning. If there is no need to change the role, then a role of the same name appears next. As time goes upward in the table, the tank has changed role two times. The initial role of the tank, Wingman, has changed to Leader as engagement started and the Leader tank was destroyed. It means the ex-Wingman tank took Leader role. But then the tank was also damaged; it transferred the role to another tank, so it became the Wingman again. The explanation for other rounded boxes is omitted because those are the issues of behavior re-planning (Kim, & Choi, 2013) and out of scope for this paper. However, the combination of re-planning and cooperation enables the Actors in simulation to be more humanlike.

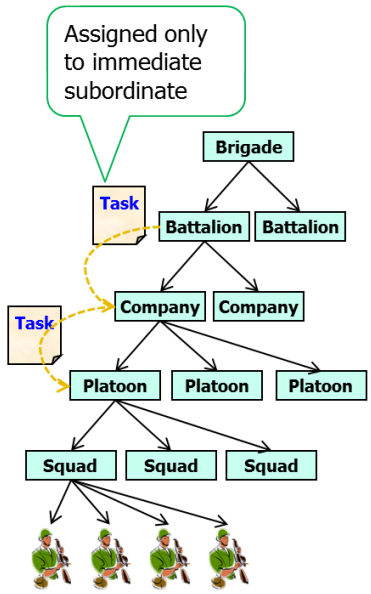


Figure 12. Scalability in ROCH

The ROCH is scalable in the resolution aspect as shown in Figure 12. It is applicable from soldier level to brigade level in that it formally and systematically specifying the tactical model of Actors from individual soldier or tank to a large scale unit (brigade) with the same scheme. Any Actor (e.g., Brigade) assigns Task to only its immediate subordinate Actor (e.g., battalions) and has no need to assign any Task to the subordinate's subordinates (e.g., companies), only to its direct subordinates. This mechanism can be applied to overall organization which is specified in scenarios. The Plan of a superior (e.g., regiment, squad) merely specifies what direct subordinates (e.g., battalions, soldiers) are required to behave.

CONCLUSION

We surveyed current outstanding war game software including OneSAF, FLAMES, and VR-Forces, and we derived a Common Model. Through the survey we figured out that existing models have scheme of tightly coupled activities (set of Tasks) with its player (Actor). Because of such coupling, it is not easy to reuse existing components (Actor) and activity definition (Tactics). Specifically it is impossible to simulate mission transfer among simulation players, which can frequently occur in real world battlefield.

To overcome such shortcomings, we have introduced ROCH model to execute the tactical models specified in the Meta-Model. It separates Tactics from its Actor, thus such loosed coupling enhances composability and reusability of the components of Actors and Tactics. The assignment mechanism of ROCH framework enables simulation to assign Roles dynamically, thus it helps simulation users to simulate more dynamically adaptable Tactics reflecting fast changing battle situation. It is also scalable from an individual soldier to brigade with the same design scheme. Thus users can specify Tactics in the same manner along the whole hierarchy.

Future Works

In implementation some exceptional cases identified. Some failures of Role assignment and re-assignment occur when an Actor loses a combat power during simulation. To solve the problem more complex plan was required.

By adding more complex conditions or conditional expressions into Plans, Plans became more complicated because of considering every exceptional case. At the initial stage of ROCH application to a simulation system, we realized that a solution is needed: Self-adaptive plan generation which enables a Unit to adaptively modify their plans according to its situation. It is possible to be applied in other domains such as Multi-Agent Systems, Pervasive Systems, and so on, to improve the composability, reusability, and adaptability of agents, components, or services.

ACKNOWLEDGEMENTS

Funding for this research was provided by the Republic of Korea's Agency for Defense Development. The support for this work is gratefully acknowledged. The authors also thank the anonymous reviewer for useful comments that helped to improve the clarity of the paper.

REFERENCES

Logsdon, J., & Nash, D., & Barnes, M. (2008). One semi-automated forces (OneSAF): capabilities, architecture, and processes. *DoD M&S (Modeling and Simulation) Conference Presentations*, Orlando, Florida, DoD M&S Conference.

Ternion Corporation. (2012). *FLAMES Simulation Framework: Online Document Version 10.0.1,* accessed on 01-07-2012, from http://www.ternion.com.

VT MÄK. (2012). VR-Forces: *Developers Guide*, accessed on 05-03-2012, from http://www.mak.com/products/simulate/computer-generated-forces.html,

Atkin, M. S., & Westbrook, D. L., & Cohen, P. R. (2001). HAC: A unified view of reactive deliberation activity, *Proceedings of the 5th International Conference on Autonomous Agents*, 92-107.

Vakas, D., & Prince, J., & Blacksten, H. R., & Burdick, C. (2001). Commander behavior and course of action selection in JWARS, *Proceedings of the 2001 Winter Simulation Conference*, 697-705.

Song, Y., & Yang, Y. (2006). Modeling organization of multi-agent system with command mechanism, *Proceedings of the 1st International Multi-Symposiums on Computer and Computational Sciences*, 732-736.

Bisht, S., & Malhotra, A., & Taneja, S. B. (2007). modeling and simulation of tactical team behavior, *Defence Science Journal, Vol. 57*, No. 6, 853-864.

Ferber, J., & Gutknecht, O., & Michel, F. (2004). From agents to organizations: an organizational view of multi-agent systems. Agent-Oriented Software Engineering IV, *Lecture Notes in Computer Science*, *Vol. 2935*, 214-230, doi: 10.1007/978-3-540-24620-6_15.

Xu, H., & Zhang, X., & Patel, R. J. (2007). Developing role-based open multi-agent software systems. *International Journal of Computational Intelligence Theory and Practice*, *Vol. 2, No. 1*, 39-56.

Cabri, G., & Leonardi, L., & Zambonelli, F. (2003). BRAIN: A framework for flexible role-based interactions in multiagent systems. On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE, *Lecture Notes in Computer Science*, *Vol. 2888*, 145-161, doi: 10.1007/978-3-540-39964-3_11.

Becht, M., & Gurzki, T., & Klarmann, J., & Muscholl, M. (1999). ROPE: role oriented programming environment for multiagent systems. *Proceedings of the 4th IFCIS International Conference on Cooperative Information Systems*, 325-333.

Hahn, C., & Madrigal-Mora, C., & Fischer, K. (2009). A platform-independent metamodel for multiagent systems. *Autonomous Agents and Multi-Agent Systems, Vol. 18, Issue 2*, 239-266, doi: 10.1007/s10458-008-9042-0.

Object Management Group. *OMG Unified Modeling Language (OMG UML), Infrastructure, Version 2.4.1*, accessed on 30-11-2012, from http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF.

Petty, M., & Kim, J., & Byun, J. (2014). Software Frameworks for Model Composition, *Modeling & Simulation in Engineering, vol. 2014*, article ID 492737.

Kim, H., & Lee, S. (2013). Role-based Command Hierarchy Model for War Fare Simulation, *International Journal of Simulation Model 12 (2013) 4*, 252-263.

Kim, J., & Choi, D. (2013). Implementation of Goal Oriented Behavior Planning, Re-planning for SAF, *Interservice/ Industry Training, Simulation, and Education Conference 2013*, Orlando, National Training and Simulation Association.