Int j simul model 12 (2013) 4, 252-263
ISSN 1726-4529 Professional paper

ROLE-BASED COMMAND HIERARCHY MODEL FOR WARFARE SIMULATION

Kim, H. S.[*] & Lee, S. W.[**]

[*] Graduate School of Information and Communication, Ajou University
[**] (Corresponding author) Department of Software Convergence Technology, Ajou University
San 5, Woncheon-dong, Youngtong-gu, Suwon-si, Gyeonggi-do, 443-749, South Korea
E-Mail: heemanz@ajou.ac.kr, leesw@ajou.ac.kr

Abstract
In warfare modelling and simulation (M&S), the mission assigned to an aggregate unit, such as platoon, company, battalion, etc., is achieved by the military operations that the unit's subordinates perform. In this procedure, the tactics for achieving the mission play an essential role for the aggregate unit to translate the abstract mission statement into concrete military operations to be performed by its subordinates in consideration of the dynamically changed warfare situation. However, the practical component-based warfare M&S software does not provide a model to explicitly represent such tactics. In this paper, we define ROle-based Command Hierarchy (ROCH) model for users to formally represent the tactics of units at design time and develop ROCH framework to support that an aggregate unit dynamically assigns roles to its subordinates considering their situation at runtime. Then, we discuss the benefits of our approach from the perspectives of composability, reusability, adaptability, and scalability.
(Received in December 2012, accepted in May 2013. This paper was with the authors 2 months for 1 revision.)

Key Words: Component-Based Warfare Simulation, Tactical Model, Dynamic Role Assignment

1. INTRODUCTION

In the warfare modelling and simulation (M&S) domain, one of traditional topics is to model and simulate the tactics for aggregate units that are composed of other units. Warfare simulations usually include a large number of diverse units. These units can be an individual unit, like rifleman, tank, aircraft, etc., or an aggregate unit, like platoon, company, battalion, etc. In these simulations, the mission given to an aggregate unit by a user can be represented to be a long-term plan that the unit should achieve in a collaborative manner. When a user writes the scenario for a simulation, the long-term plan implicitly reflects the high-level tactics that the unit should conform to. These tactics are realised in the simulation world through the military operation that the unit's subordinates execute. In this procedure, tactics play an essential role for the warfare M&S software to interpret users' intention (i.e. the mission of a unit) into the unit-executable military operations.

Recently, there are several on-going researches that apply component-based software engineering to the development of warfare M&S software, such as OneSAF [1], FLAMES [2], and VR-Forces [3]. In this software, a unit is usually represented to be an instance of a unit component. This software helps developers to develop M&S systems fits to users' purpose with low cost using the reusability of components. Consequently, users can rapidly obtain the M&S system they need. However, this software does not yet provide developers and users with a model to explicitly represent the tactics for units. Based on this model, developers can systematically make reusable unit components able to perform diverse tactics as well as users

can flexibly represent the tactics for units proper to their purpose. In addition, developers and users can share the understanding of the tactics represented in this model.

In the previous work related to tactical model, researchers have proposed several models, such as hierarchical agent control (HAC) [4], commander model (CB) [5], command-based multi-agent system (CMAS) [6], tactical team behaviour (TTB) [7], agent-group-role (AGR) [8], and so forth. Each model is helpful to represent the tactics of aggregate units and to enable the units to perform the tactics. However, the tactics in these models tend to be tightly coupled with units that can perform them. Therefore, it makes the reuse of units or tactics difficult.

In this work, we propose ROle-based Command Hierarchy (ROCH) model and framework to overcome the shortcomings of the aforementioned work. ROCH model is used to formally represent the tactics of aggregate units for the purpose of separating the tactics from unit components. In order to achieve these goals, we bring the concept of role into our model. Within the model of an aggregate unit, roles logically connect tactics with its subordinates at design time under the situation where tactics and the subordinates are separated. ROCH framework is used to dynamically bind the aggregate unit's subordinates with roles considering their situation at runtime. Consequently, this work enables units to be more composable, reusable, and adaptable than those of the previous models.

The remainder of this paper is organised as follows. We define a common model of the existing component-based warfare M&S software in Section 2. Based on the common model, we propose ROCH model and framework in Section 3. In Section 4, the advantages of ROCH model and framework are described through a case study. Finally, we conclude and discuss this work in Section 5.

2. A COMMON MODEL FOR WARFARE M&S SOFTWARE

There are common modelling elements to represent the units in the existing component-based warfare M&S software products. At first, we summarize these modelling elements from OneSAF, FLAMES, and VR-Forces. That are widely used for military M&S. Table I shows the summarised modelling elements in each software product. The last CSFA, which is the abbreviation of Composable Simulation Framework Architecture, is our simulation software that we are currently under development. At a glance, there are some differences between the modelling elements for each product, but many of them are used as similar concepts. These similar modelling elements are only called in different names for each product as shown in Table I.

Table I: Summary of modelling elements for units in warfare simulation.

Modelling Elements	OneSAF	FLAMES	VR-Forces	CSFA
Unit	Actor	Unit	Entity	Unit
Stand-alone Unit	Entity	Unit	Entity	Primitive Unit
Composite Unit	Unit	-	Aggregate Entity	Aggregate Unit
Equipment	Physical Agent / Physical Model	Equipment	Sensor / Actuator	Equipment Agent / Equipment Model
Behaviour Logic	Behaviour Agent / Behaviour Model	Cognition Model	Controller	Behaviour Agent
Intra Unit Communication	Trigger	Specific interface	Port	Trigger
Inter Unit Communication	Event	Message	Message	Event
Cooperation	Behaviour Agent	Cognition Model	Controller	Behaviour Agent

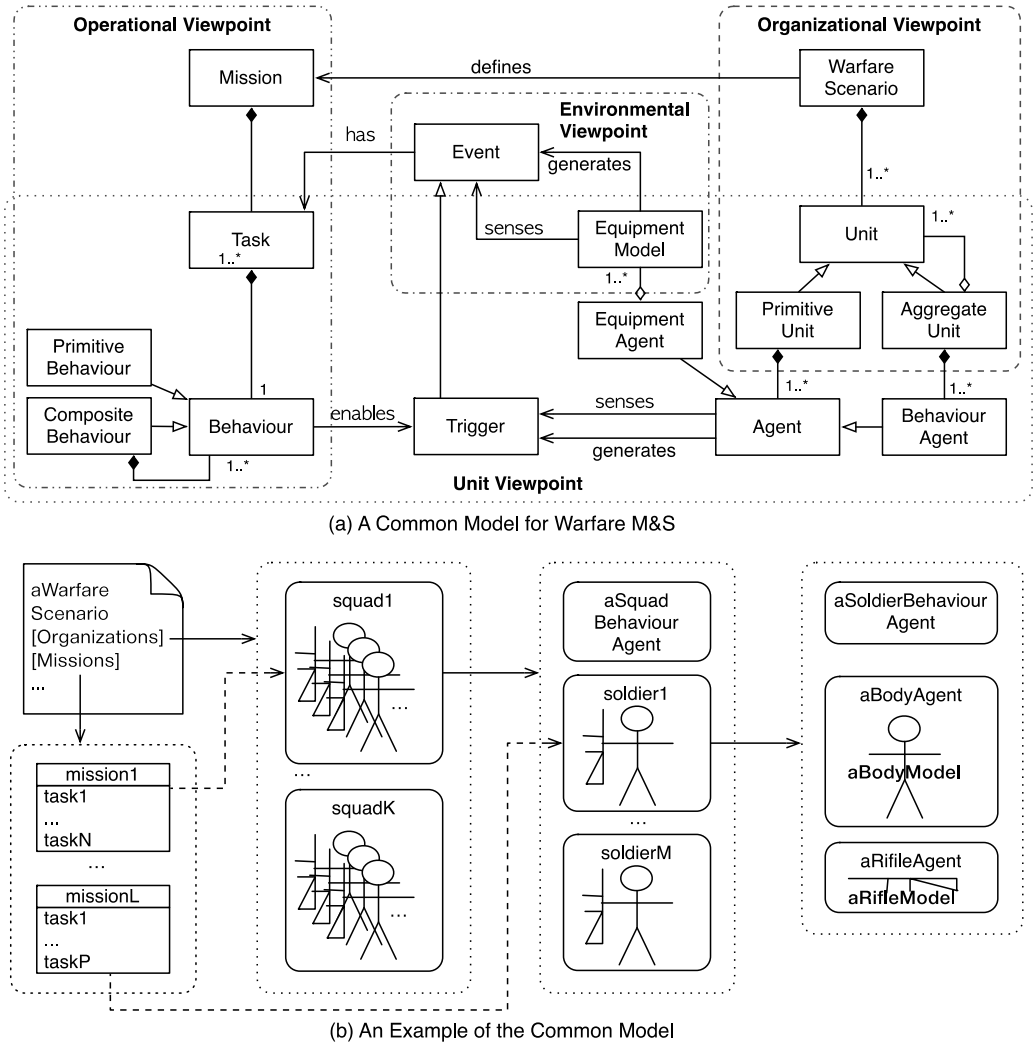

Figure 1: A common model and an example for warfare M&S.

From the observation of the existing products, we define a *common model* as the preliminary work for the modelling of aggregate unit's tactics. The common model is represented in Fig. 1 using the terms of CSFA. Fig. 1 (a) shows the overview of the common model, and Fig. 1 (b) provides an example to help the understanding of this model.

From the environmental viewpoint, a unit can detect or make a change of the environment around them through the equipment model(s) of their equipment agents. This change is delivered to other equipment models in the simulation environment in the form of an event. For example, a fire event generated by the rifle model of a soldier in Fig. 1 (b) is delivered to a body model of other soldier to calculate its damage effect. An event may also contain a task needed by a superior unit or a user. The dash allows in Fig. 1 (b) show the tasks initially assigned by a user in a warfare scenario.

A warfare scenario usually includes the organisational information for each side like blue or red force. From the organisational viewpoint, this information is represented as a hierarchy that consists of units. An aggregate unit consists of other units while a primitive unit does not include any unit. Subordinate relations between aggregate units and their subordinates reflect a chain of command in military domain. In Fig. 1 (b), the squads and soldiers are aggregate units and primitive units, respectively.

A unit is composed of agents from the unit viewpoint. This concept is borrowed from OneSAF [1]. An agent is specialised into an equipment agent or a behaviour agent. An

equipment agent controls its equipment model to represent human, tank, radar, flight, rifle, etc. A behaviour agent decides the behaviour of a unit in the current situation. Agents within a unit can communicate with others using a trigger, which is a kind of event. In addition, a unit implicitly has the tasks that it can achieve. A task is related with a unit through the behaviour, which is included in the task, and the triggers, which is enabled by the behaviour. A primitive unit can be composed of equipment and behaviour agents, and an aggregate unit can be composed of only behaviour agents and other units. For example, a soldier consists of a body, rifle, and soldier behaviour agents while a squad consists of multiple soldiers and a squad behaviour agent in Fig. 1 (b). In this figure, we skipped the tasks able to be performed by each unit for simplicity.

From operational viewpoint, a mission is a conceptual goal that a unit has to achieve in a simulation. A mission can be composed of tasks that are arranged in order. This series of tasks means long-term plan in the position of the unit that performs the mission. Tasks are delivered to a unit in the form of an event as abovementioned. A task is specified as a combination of behaviours. Behaviour is specialised into composite or primitive behaviour. A composite behaviour is composed of other behaviours that are arranged in order. A primitive behaviour can cause the unit's behaviour agent to generate triggers. The unit's agents can react on the triggers that they are interested in. Based on this mechanism, a unit completes the tasks assigned to it.

These modelling elements during simulation operate as follows. When starting a simulation, each unit receives a series of tasks for achieving a mission of the simulation through its communication model that is one of equipment models. The equipment agent with this communication model delivers these tasks to the behaviour agent. The behaviour agent translates tasks into behaviours and issues the triggers required by each behaviour. Other agents of the unit react to the triggers. Independently of this procedure, the equipment models of the unit can sense or generate events like fire, location update, etc. The event sensed by a unit can affect the behaviour of the unit. These two procedures are repeated until the simulation is finished.

3. ROLE-BASED COMMAND HIERARCHY

ROCH is described on top of the common model described in Section 2. In order to separate tactics, which are composed of tasks, from units that perform the tasks, we bring the concept of a role into ROCH. Fig. 2 shows the main idea of ROCH.

The previous models, like HAC [4], CM [5], and TTB [7] focus on modelling of tactics without consideration of unit components' composability. On the other hand, the previous role-based models, such as CMAS [6] and AGR [8], 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 that the units can play. When developing warfare M&S software with these models, it is easy that the units are tightly coupled with their tactics. The result of this coupling can be represented as shown in Fig. 2 (a). This means a superior unit should be dependent on its subordinates or vice versa.

In order to overcome the drawback of this tight coupling, ROCH uses roles as the abstract unit to indicate an actual subordinate unit, which is not predetermined at design time. And then each role is bound to an actual unit at runtime as shown in Fig. 2 (b). In ROCH, the tactics to achieve each task assigned to an aggregate unit is represented as a plan. Therefore, aggregate units have a plan for each task. The plan makes the subordinates of a unit tactically behave. For representing this, a plan is represented with roles and a series of sub-tasks. A role has a set of sub-tasks that its player should be able to achieve. A series of sub-tasks is the

tactical procedure for the aggregate unit to achieve the given task. A sub-task indicates the task to be achieved by the player of a role at runtime.

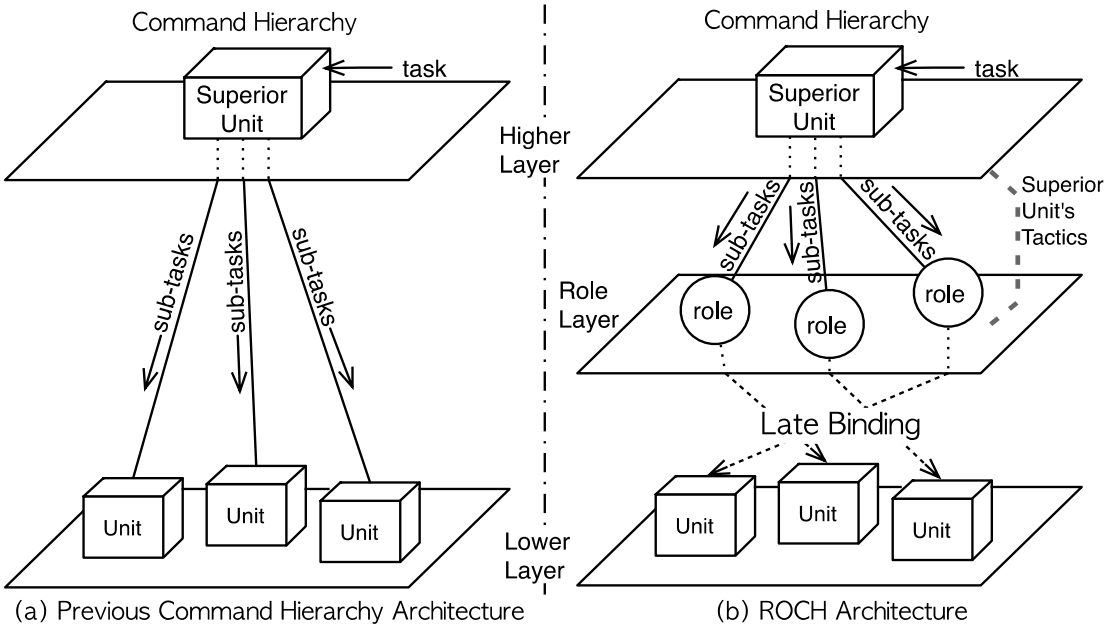

Figure 2: The concept of ROCH.

Unlike aggregate units, primitive units do not have any plan in ROCH because they cannot have any subordinate. Primitive units merely perform a given task by themselves according to the common model described in Section 2. Even though the architecture of ROCH in Fig. 2 shows only two-level hierarchy, it can be applied to the overall hierarchy on a side of a warfare simulation in the same manner. The details are discussed in Section 4.4.

For the implementation of this concept, ROCH is realised into two artefacts: a meta-model and a framework. The meta-model is used to specify units and plans to be a machine-readable representation using XML, differently from the previous role-based approaches [6, 8-12] where roles can be tightly coupled with their player at development time. The proposed meta-model allows the tactics of aggregate unit to be freely modified by users for their simulation purpose. The framework provides the facilities for each aggregate unit to execute their plans with a mechanism to dynamically assign roles in consideration of its situation at runtime.

3.1 Meta-model for ROCH

The proposed meta-model in the context of the model-driven architecture [13] is a language to represent a unit from the unit and tactical viewpoints. Fig. 3 shows the ROCH meta-model.

In the unit meta-model, a unit has attributes and tasks (capabilities). The attributes of a unit are used as variables to represent the state of the unit. The tasks of a unit mean that the unit can perform the task.

In the tactical meta-model, the plans for each aggregate unit are specified. These plans are matched with each task that the aggregate unit can accept. A plan is composed of roles and a plan expression. A role has attributes, tasks (responsibilities) and two conditions (assigning and withdrawing conditions). The attributes and tasks of a role can be considered to be a kind of requirement. For a subordinate unit to take the role, it must have the attributes and tasks that are matched with the attributes and tasks of the role, respectively. It assures that a subordinate, which are playing a role, can perform sub-tasks specified in the role as well as

the superior can determine the current situation from its subordinates' state because they already share the semantics of tasks and attributes. Two conditions are specified in a logical expression that uses attributes of the role as variables. The conditions of a role are used for superior unit to dynamically determine the player of the role among their subordinate units. The assigning condition of a role represents the condition in which a subordinate can play the role. The withdrawing condition of a role represents the condition in that a subordinate, which is playing the role, cannot play it anymore. Comparing with the previous models, such as OneSAF [1], FLAMES [2], and VR-Forces [3], without the support of dynamic role assignment, these two conditions in ROCH model enable an aggregate unit to dynamically assign roles to the proper subordinates considering their subordinates' dynamic situation at runtime. A plan expression represents a series of the sub-tasks to be achieved by the player for each role. This series of sub-tasks is represented to be a sequential, concurrent, conditional, or role-task expression. A sequential, concurrent, or conditional expression can have other expressions. A role-task expression is used as the terminal expression to represent a task that a role's player should perform.

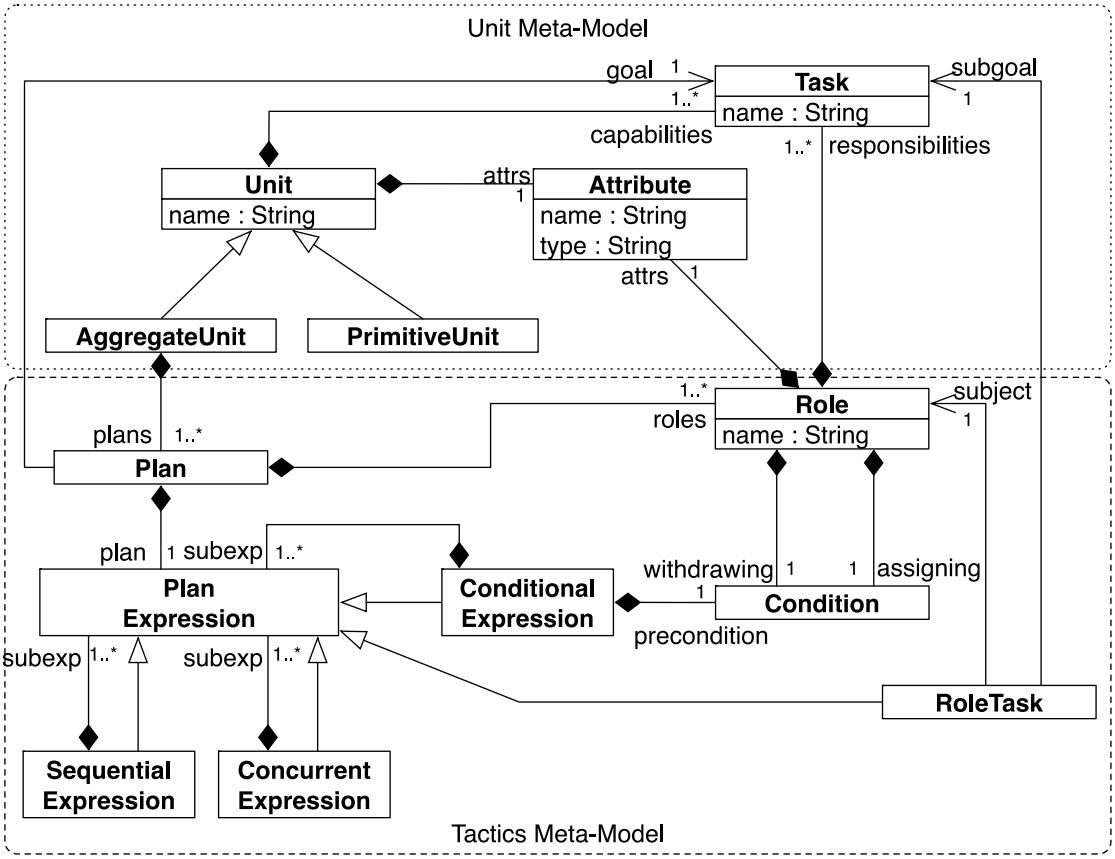

Figure 3: ROCH meta-model.

3.2 Framework for ROCH

The main function of ROCH framework is to execute the tactical model (i.e. plans) under the behaviour agent of units. This framework is composed of *Plan Lib*, *Role Manager*, and *State Manager*, as shown in Fig. 4. *Plan Lib* loads the plans for an aggregate unit. Also, it selects and activates the plan for the task received from a user or a superior unit. *Role Manager* assigns/withdraws the roles in the activated plan to/from the aggregate unit's subordinates according to assigning/withdrawing conditions. *State Manager* maintains the states of its owner unit and the subordinates. The followings describe the operation of ROCH framework.

ROCH framework operates based on the information from a scenario and unit components for a warfare simulation as represented in top of Fig. 4. A scenario includes the organisational information like that *Unit A* is a superior of *Unit B*. The scenario has references to the configuration of unit components. The configuration of a unit component includes the unit model, which is specified in unit meta-model of ROCH meta-model, and an implementation package. If the unit component is for an aggregate unit, the configuration additionally refers to the plan for each task that the unit can perform.

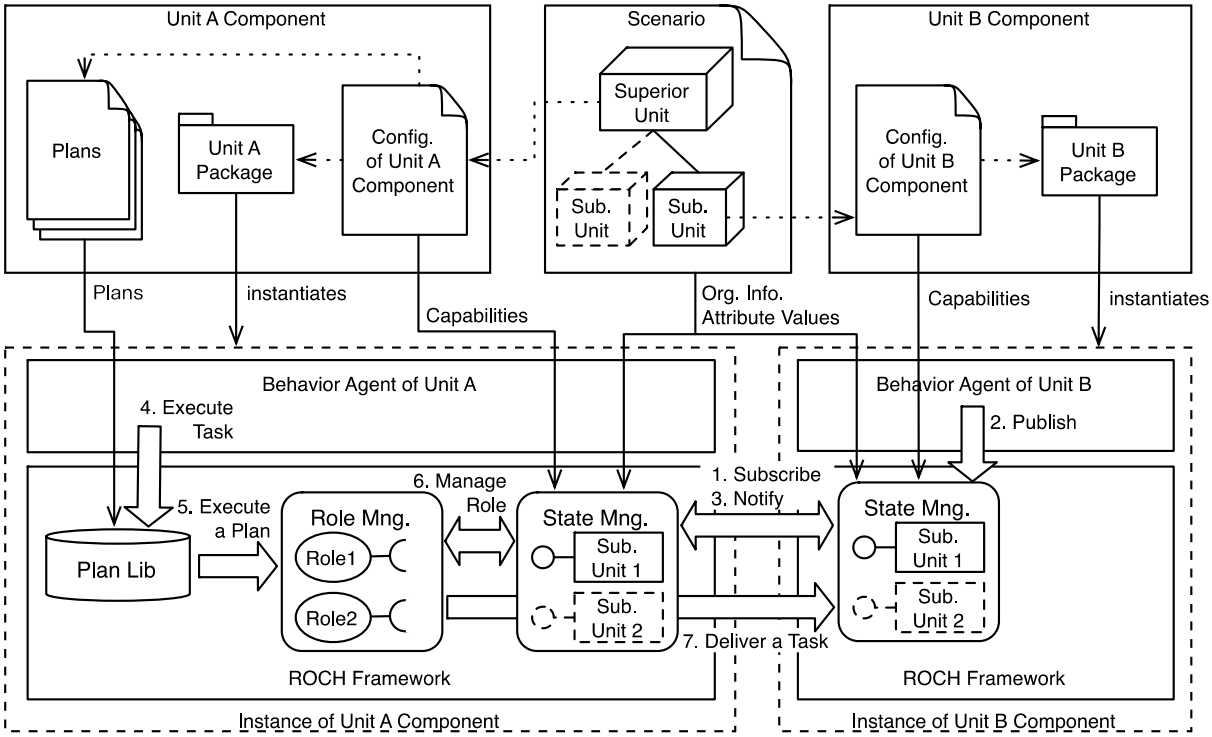

Figure 4: ROCH framework.

The function of ROCH framework is separated into two sub-functions: state informing function and plan execution function. The first function is for a unit to inform its superior unit about its own capabilities and attributes. It is called when the unit participates in its superior, or the capabilities and attributes of the unit are changed. This is designed using publish/subscribe pattern (1~3) between the *State Manager*s of a unit and its subordinate. The information shared by this function is used by the following function.

The second function is to execute the plan for achieving a task assigned by a user or its superior unit. When an aggregate unit receives a task (4), the *Plan Lib* of the unit selects and activates the plan for it (5). Then, *Role Manager* assigns the proper subordinates the roles specified in the plan (6). After assigning roles, each role is bound with a subordinate. Then, a series of sub-tasks are delivered to role-playing subordinates (7). Each subordinate performs the received sub-tasks. If a role-playing subordinate cannot play the role (i.e. the subordinate's state meets the withdrawing condition of its role), the role is reassigned to one of possible subordinates by *Role Manager*.

3.3 Procedure for role management

ROCH framework supports a dynamic role assignment at runtime using the procedure shown in Fig. 5. The behaviour agent of an aggregate unit periodically calls *Update* function of ROCH framework (1.1). At this time, the framework checks if there are roles to be withdrawn

from/assigned to its subordinates using roles' *withdrawing/assigning conditions* described in Section 3.1 (1.2/1.3). In these steps, the information of roles and subordinates obtained by *State Managers* is used. In the withdrawing step, the framework checks whether every assigned role's withdrawing condition is met by the current state of the subordinate playing the role or not. If the condition is met, the assigned role is withdrawn. After checking role withdrawing, released roles are assigned to subordinate units according to a mechanism. The flow chart in the right of Fig. 5 shows the mechanism of assigning a role to a subordinate in this work. For a subordinate to play a role, its capabilities and attributes should cover the responsibilities and attributes of the role, respectively. As explained in Section 3.1, the capabilities of a unit are its achievable tasks, and responsibilities of a role are its required tasks. The last step is to assign a role to a subordinate if the assigning condition of the role is met by the current state of the subordinate. After role assignment, a pair of the role and the subordinate is created in the framework (1.4).

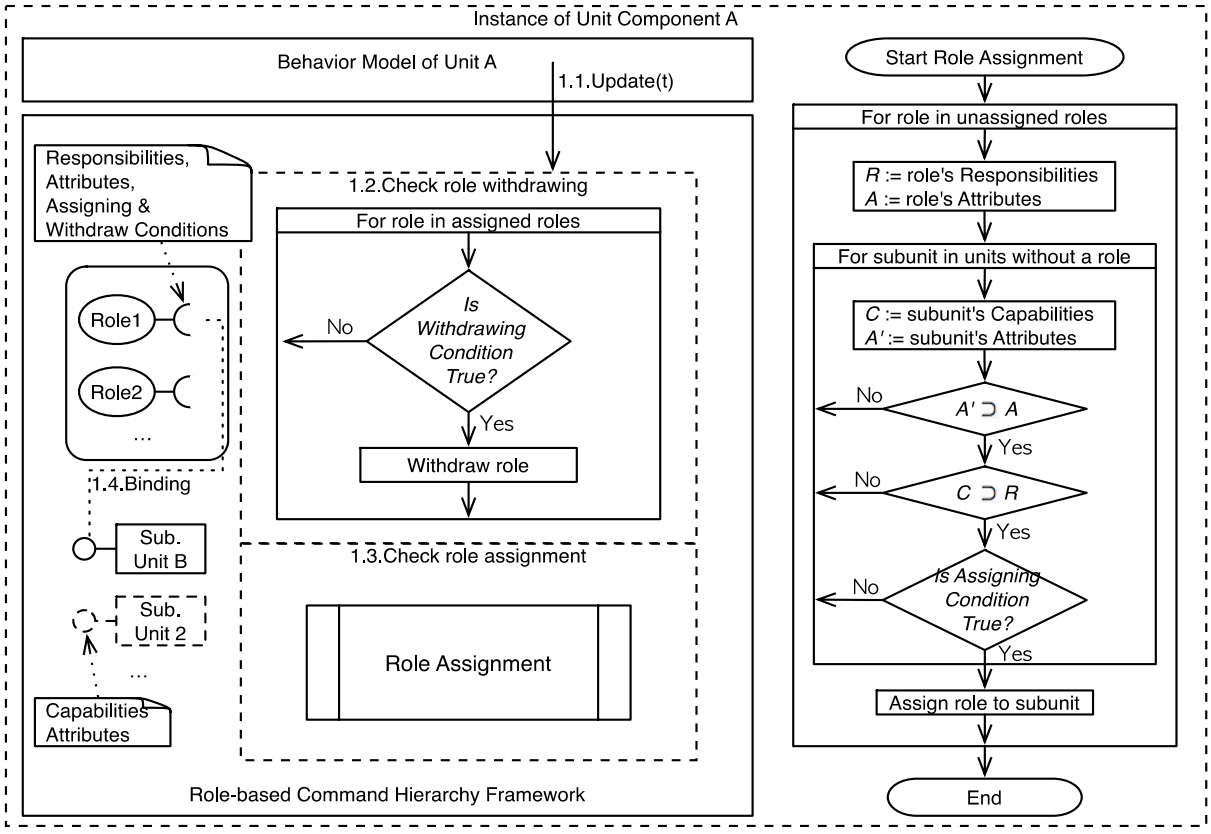

Figure 5: Role manager's function.

4. A SCENARIO-BASED CASE STUDY

In this section, we show how ROCH model and framework improves the composability, reusability of unit component, the reusability of tactical model, and the adaptability of unit. Additionally, we discuss the scalable application of ROCH model. In order to demonstrate these benefits, we firstly introduce a simple combat scenario and then specify the units' tactical models required to simulate this scenario. Then, we discuss some cases where the benefits of ROCH model are explicitly exposed.

Suppose that a user should simulate the combat scenario in which an infantry regiment of blue side engages with a battalion of red side to occupy *Point X* as shown in Fig. 6 (a). For simplifying this discussion, we make some assumptions described in Fig. 6 (b). Based on

these assumptions, we define two unit models for regiment (c) and battalion (d), and the plan for a regiment to occupy a location (e). This scenario is very simple, but it contains the context needed to explain the benefits of ROCH discussed in the following sections.

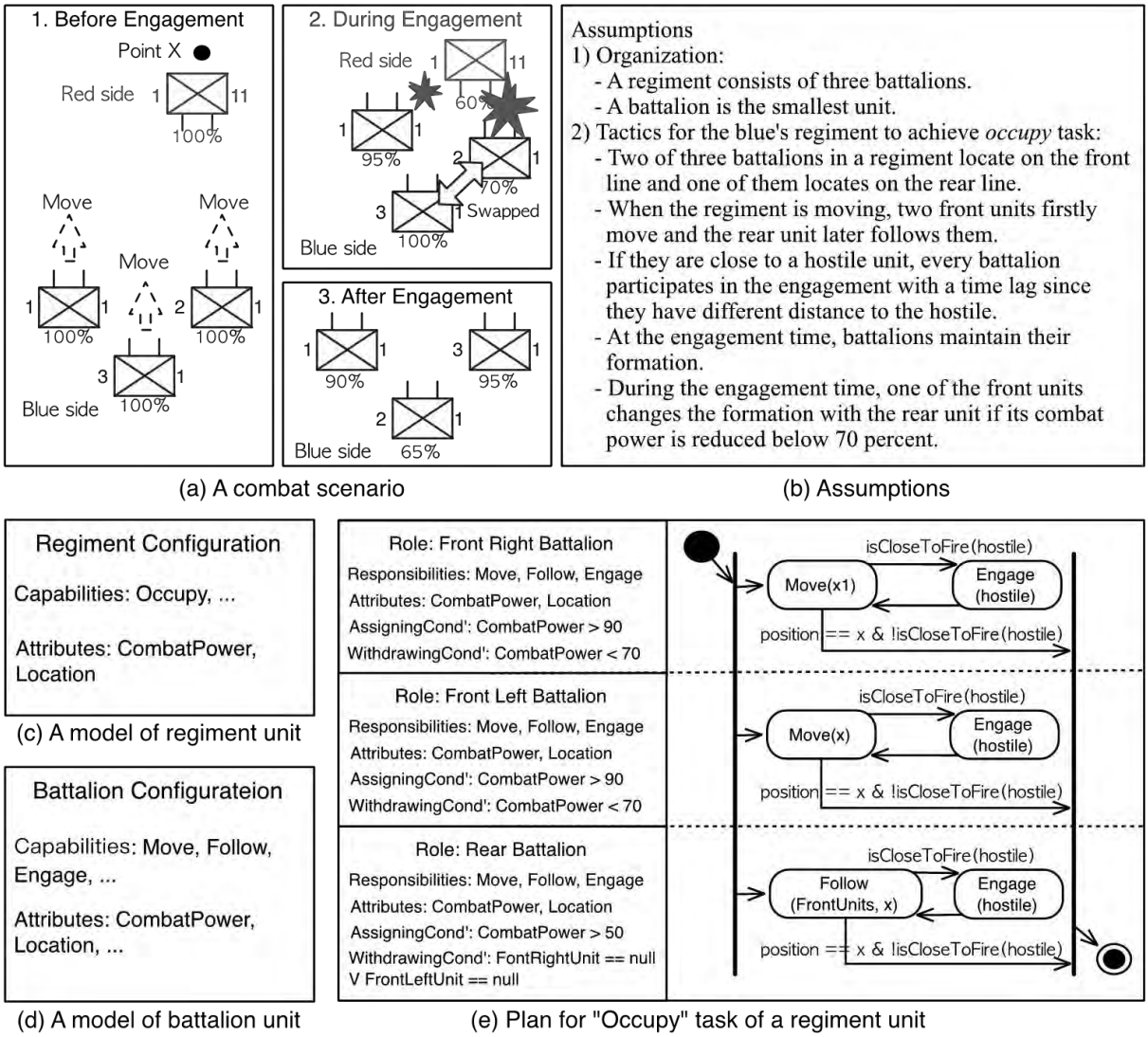

Figure 6: Example of combat scenario and the models for units and tactics.

4.1 Composability of unit component

ROCH model improves the composability of unit components. For a newly developed unit component to interact with pre-existing unit components, the new component should be developed dependently on the pre-existing components. In Fig. 7, case 1 shows an example of which a new unit component (Component B) is directly dependent on a pre-existing unit component (Component A). Case 2 shows that two unit components (Component A and Component B) are combined through an interface for loose coupling. However, a problem occurs when later developed unit component (Component K) needs to be combined with several existing unit components (Component A, Component B, etc.) as shown in case 3. The later developed unit component must implement all interfaces required by the existing unit components to be combined. It may also cause a multiple inheritance problem.

In order to overcome this problem, ROCH model represents each tactics for an aggregate unit as a plan specified with the tasks that subordinates can understand and perform. See case

4 in Fig. 7. The instance of a new unit component (Component B) can be a subordinate of a unit component (Component A) if the new unit component has the attributes and tasks required by role definitions in its superior component. Similarly, it is possible for the new unit to be a superior of other units. This makes the composition between unit components more flexible. In Fig. 6, there is no direct dependency or inheritance between regiment and battalion components. The capabilities of battalion configuration in Fig. 6 (d) are merely used as roles' tasks to specify the plan of regiment component like Fig. 6 (e).

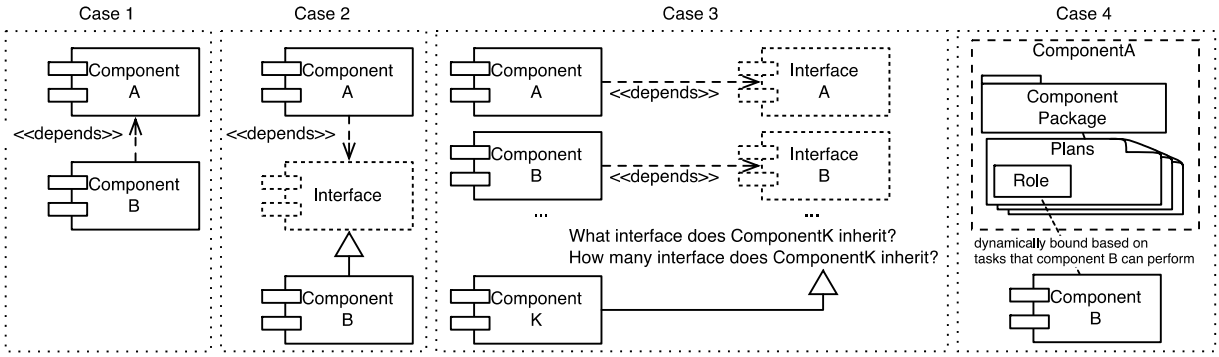

Figure 7: Composition of components.

4.2 Reusability of unit component and tactical model

ROCH model improves the reusability of unit components and their tactics. Imagine the case when a user edits the scenario of Fig. 6. There are two possible variances that ROCH model allows. One is to modify the tactics of a superior unit component like the regiment component in Fig. 6. In this case, the user can reuse the existing subordinate component, such as the battalion component in Fig. 6, without rebuilding if the user does not require that the battalions perform new task that they cannot understand. Other is to replace a subordinate unit component, like the battalion component that is located on the rear line in Fig. 6 (a), with a new type of battalion like an armored battalion. In this case, the user does not need to modify the regiment component's plans if the new battalion can understand the tasks required in the plans. Even though this reusability has the obvious limit, our model can maximize the reusability of existing unit components under the limit because it allows the tactics of a superior unit to be defined using several combinations of the tasks shared between the superior and its subordinate units.

4.3 Adaptability of unit

In warfare M&S software, units are often located in the unexpected situation, such as the loss of resources, the encounter of unknown enemy, etc. As one of ways to solve this situation, ROCH provides some adaptability to aggregate units. The adaptability is to reorganise the subordinates within aggregate units considering their dynamically changed situation at runtime. In ROCH model, each role specified in an aggregate unit has assigning and withdrawing conditions to specify the conditions where role is assigned to a unit and withdrawn from its player unit, respectively. The ROCH framework of the aggregate unit assigns/withdraws the roles to/from subordinate units based on these conditions at runtime. It makes each subordinate differently behave according to the role that it is playing and its state. For example, the definition of roles in Fig. 6 (e) explains that one of two front battalions will swap seats with the rear battalion if its combat power is damaged below 70 % at runtime. This tactical behaviour of the regiment is initiated by the ROCH framework of the regiment as

follows: 1) withdrawing the *front right battalion* role and the *rear battalion* role from their players, 2) reassigning the withdrawn roles to proper battalions, and 3) delivering 'Move' and 'Follow" tasks, which are shown in Fig.6 (e), to the reassigned role's players, respectively. As this result, the front right battalion moves behind other two battalions.

4.4 Scalable application of ROCH model

In this section, we discuss the range to which ROCH model can be applied. Users or developers can formally and systematically specify the tactical model of a unit from individual soldier to corps in ROCH meta-model. Assume that company is the smallest unit in the scenario of Fig. 6. Therefore, we need a company component and the tactical model for each battalion to control companies of which it is composed. The plans of the regiment component merely specify what battalions are required and how they tactically perform the assigned tasks to the regiment. The regiment does not directly assign any task to the companies of its battalions. The tactical model to control the companies is specified in the plans of the battalion component. This mechanism can be expanded to an overall organisation, which is specified in simulation scenarios.

5. CONCLUSION

In this paper, we proposed ROCH model and framework to specify and simulate the tactics of aggregate units in composable warfare M&S software. Firstly, we analysed the previous component-based warfare M&S software, such as OneSAF, FLAMES, and VR-Forces. Through this analysis, we derived a common model able to be used in the warfare M&S software. Next, we propose a meta-model for ROCH on top of this common model and design ROCH framework to execute the tactical models specified in the meta-model. In order to design ROCH meta-model, we have identified a main element (i.e. task) used for the interaction between an aggregate unit and its subordinates on the chain of command. Then, roles and plans are defined with the element in the unit-understandable description level. At runtime, The ROCH framework of aggregate units assigns the roles, which are defined in the model of them, to proper subordinates by considering their state and capabilities.

Our approach provides users and developers with the following benefits. Firstly, ROCH model and framework provide developers with the composability and reusability of unit components and users with the reusability of tactical models. It is technically achieved by the separation of its tactical model from unit component. Secondly, the model and framework enable aggregate units to adapt the tactical behaviour at runtime using the proposed role assignment mechanism. It helps users to simulate more dynamically adaptable tactics of units onto the change of simulation environment. Finally, developers can develop unit component from an individual unit to corps with the same design scheme based on ROCH model and users specify the tactics for these units in the same manner. This benefit is originated from the scalable modelling scheme of ROCH model.

Presently, we are developing the component-based warfare M&S software including ROCH model. In the development, we have identified that the developed unit component at runtime frequently meets the exceptional cases: the failure of role assignment during simulation and the occurrence of role re-assignment during executing a plan when a unit loses a combat power during simulation. Even though ROCH model can deal with these cases by adding conditions and conditional expressions into plans, these counter measures require the plans to be more complex. It means that users have to consider every exceptional case. Exceptional cases are usually issued when a unit meets the unpredictable situation at the modelling time. However, it is difficult that users specify the plans for a unit considering

every situation in which the unit can be located. A solution for this is to enable an aggregate unit to adaptively modify their plans according to its situation. In future work, we have a plan to invent a self-adaptive plan generation for aggregate units on the basis of ROCH model.

The proposed approach in this work is merely applied to composable warfare M&S domain, but we expect that this can be widely applied to several domains, such as Multi-Agent Systems, Pervasive Systems, and so on, to improve the composability, reusability, and adaptability of agents, components, or services.

6. ACKNOWLEDGEMENT

This research was supported by the Agency for Defense Development under the contract No.UC100007ID in conjunction with REATIMEVISUAL Co. Korea and Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (2013M3C4A7056233).

REFERENCES

[1] Logsdon, J.; Nash, D.; Barnes, M. (2008). One semi-automated forces (One SAF): capabilities, architecture, and processes, *DoD M&S (Modeling and Simulation) Conference Presentations*

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

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

[4] 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

[5] 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

[6] 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

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

[8] 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

[9] 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

[10] 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

[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

[12] Hahn, C.; Madrigal-Mora, C.; Fischer, K. (2009). A platform-independent metamodel for multiagent systems, *Autonomous Agents and Multi-Agent Systems*, Vol. 18, No. 2, 239-266, doi:10.1007/s10458-008-9042-0

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