Journal Title
XX(X):1–14

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Modeling the Behavior of a Hierarchy of Command Agents with Context-based Reasoning

Rikke Amilde Løvlid[1], Solveig Bruvoll[1], Karsten Brathen[1], and Avelino Gonzalez[2]

Abstract
Context-Based Reasoning is a paradigm for modeling agent behavior that is based on the idea that humans only use a small portion of their knowledge at any given time. It was specially designed for representing human tactical behavior and has been successfully implemented in systems with single agents or two agents working together. In this paper we apply this idea in a hierarchical multi-agent system of command agents, where the agents' actions are to command and coordinate subordinates, send reports to their superiors, and communicate with other agents at the same level. We focus on how contexts and actions can be defined for these higher level command agents and how the contexts and actions for the different command agents are related. The proposed methodology is implemented and tested for a hierarchy of command agents that are interpreting and planning an operational order at a battalion level and carrying it out in a computer generated forces environment.

Keywords
Computer generated forces, behavior modeling, multi-agent system, simulation

1 Introduction

Battle management is the planning, coordination and monitoring conducted by leaders of a military operation[1]. At each level of the military hierarchy a commander and his staff receive a task from their superior, plan how the task should be executed, and command their subordinates to carry out the plan. The subordinates of a commander are his available resources, and the planning process involves how these resources should be allocated to fulfill the task. The commander gives orders to his subordinates and monitors the battle based on reports from the subordinates. A change of the situation may require the commander to take action, e.g., if units become lost or unexpected enemies are encountered, the commander may need to reallocate resources and assign new tasks to the subordinates. If he realizes that he cannot fulfill his mission, he needs to report this to his superior.

Intelligent agents that can simulate battle management can be useful both for command and staff training and for operational planning. Current command and staff training uses simulation systems that consists of computer generated forces (CGFs) in combination with human operators. The human operators receive high-level tasks (e.g., company level) as input, transform these into lower-level tasking for subordinate units (e.g., platoon level and lower), and then they manually enter the more detailed sets of instructions into the simulation system. A challenge in the case of training is that the large amount of resources required inhibits frequent training events. The need for a large simulation support staff is even more problematic if simulations are to be used more during operations, in planning or mission rehearsal, e.g., for what-if analysis. We therefore investigate how we can make more autonomous and intelligent CGFs, by incorporating a representation of the battle management organization and its decision making. This is realized by building a multi-agent system of command agents that control a commercially available CGF system.

Our multi-agent system consists of a hierarchy of command agents. One agent is created for each unit in the order of battle of the military force being simulated. Each agent represents the leader of that unit and his staff. When an agent receives a task, it decomposes that task into tasks for its subordinates. The lowest level of our command agents then assigns tasks to units in a commercial CGF system, which then execute the tasks in the simulation.

[1]FFI - Norwegian Defence Research Establishment, Norway
[2]University of Central Florida, USA

Corresponding author:
Rikke Amilde Løvlid, FFI, P.O.Box 25, 2027 Kjeller, Norway
Email: rikke-amilde.lovlid@ffi.no

The decision making of our command agents is modeled with Context-based Reasoning (CxBR). The objective of this paper is to describe and discuss how we implemented this application with CxBR.

CxBR is a paradigm for modeling human tactical behavior based on the idea that humans only use a small portion of their knowledge at any given time [2]. The specific knowledge required is indexed according to the context the agent would face, and its use is triggered by the agent's recognition of the appropriate context. CxBR encapsulates knowledge about appropriate actions, procedures, and expectations as well as possible new situations into modules called *contexts*. Thus, which actions an agent can select and what sensory input it should consider depend on its current context. The appropriate context is determined based on the overall goal of an agent and the current situation.

CxBR has been developed at the University of Central Florida (UCF) during the last 20 years, and has been successfully used for agent behavioral modeling in several applications. Automobile driving has often been used as an example application [3–5], and Patz et al. used CxBR to control an autonomous robot car in the DARPA Grand Urban Challenge of 2008 [6]. CxBR has also been used to model a submarine's tactical mission [7;8], in applications for naval surface ships [9;10], to model autonomous helicopters [11] and for smart buildings [12]. Several of these applications used a CxBR Framework developed by Norlander to provide an infrastructure optimized for representing human behavior in CxBR [13].

Most of the work on CxBR has focused on the control of task execution for one agent, but extensions of CxBR to handle collaborative behaviors and teamwork have been addressed by Devero [14], and Barrett and Gonzalez [15]. Grama has used the concept of CxBR for representation of tactical planning for mid-echelon Army applications (battalion, company, etc) [16].

Traditionally, both the behavior logic inside each context and the knowledge of choosing the appropriate context have been modeled "by hand", i.e. by hand coding rules or writing scripts. We have done the same in our multi-agent system, but there are also promising results on using learning from observation together with CxBR [17–21].

CxBR introduces several advantages as a way to model human behavior (actions) in tactical operations. The most significant one is that it can encapsulate the knowledge required to manage certain situations recognized by the agent and make it accessible in an efficient manner. Secondly, its intuitive nature, as reported by Gonzalez [2], can make novice developers quickly adapt at building applications. Thirdly, the distributed and modular nature of CxBR can facilitate the expansion of the spectrum of behaviors to be displayed by the agent.

CxBR is different from meta-rules and finite state machines in that a context seeks to contain all that is relevant when in that context. As stated in [2], a context " ... knows what to do, how to do it, and what to expect ... " when in a specific situation.

This paper focuses on how CxBR can be applied to represent the decisions and actions of command agents, and extends the application of CxBR in several ways. The main contribution is an application of CxBR to a multi-agent, hierarchical management problem as opposed to a single agent execution problem. This introduces a number of new challenges such as "What are the contexts of a command agent?" and "How are contexts of command agents in a hierarchical multi-agent organization interrelated?".

The concept for using CxBR to represent a hierarchical management organization is implemented and tested in a battalion operation as a part of the work described here. This use case includes a mid and lower echelon battle management organization consisting of the battalion, company and platoons levels. These levels are represented by a battalion command agent superior to company command agents with subordinate platoon command agents for each company command agent. Additionally, the use case includes executing agents in the sensor-controller-actuator architecture of the commercial CGF system discussed earlier, which control main battle tanks and infantry fighting vehicles.

The paper is organized as follows. In the next section we provide some necessary discussion on CxBR and intelligent agent modeling. In section 3 we elaborate on the application of battle management organizations and how we have applied CxBR to represent them. Section 4 outlines the battalion operation which we have used as a use case to test our developed concepts. Results and experiences are discussed in section 4. A discussion of the results is included in section 6, before we conclude and suggest further work in section 7.

2 Intelligent Agents and Context-Based Reasoning

An *agent* is an autonomous entity that observes through sensors and acts upon its environment using actuators in order to meet its mission objectives. To be called intelligent, according to Wooldridge, an agent also has to be reactive, proactive and social; meaning, it must be able to react to changes in its environment, pursue goals, and communicate with other agents [22]. Russel and Norvig

defines an intelligent agent as one displaying rational or human like behavior[23].

A multi-agent system (MAS) consists of a number of intelligent agents that communicate with each other directly or through their interaction with the environment. The agents in a multi-agent system can be identical (homogeneous MAS) or different (heterogeneous MAS), and they can be cooperative or self-interested. The term multi-agent system often mean a group of software agents, but can also be used for a group of robots. Note that a multi-agent system and an agent-based system is not the same. The term agent-based system typically mean a group of simple, non-intelligent agents.

CxBR can be used to represent the behavior of intelligent agents and was especially designed to represent human tactical behavior, i.e. behavior consisting of a time series of decisions to reach a goal. The idea is to divide the knowledge into contexts in order to limit the number of possibilities for the action selection process. An extensive description of CxBR can be found in[2].

In CxBR the contexts are organized in a context hierarchy consisting of a mission context, and major and minor contexts. The *mission context* is a purely descriptive context, meaning it does not describe behavior nor is used to control the agent. A mission context contains a goal and a *context plan* for reaching it together with parameters and a *context map*, as illustrated in figure 1. The context plan is a sequence of major contexts with defined transition criteria. The context map defines all possible transitions between the major contexts available for the mission at hand and parameters. For a military operation application examples of parameters are objective areas, phase-lines, routes, etc.

Major contexts constitute the next level in the context hierarchy and are the controlling element for an agent. There is only one major context in control of an agent at any time, called the *active context*. A major context basically contains three kinds of knowledge: *action knowledge*, *transition knowledge* and *declarative knowledge*.

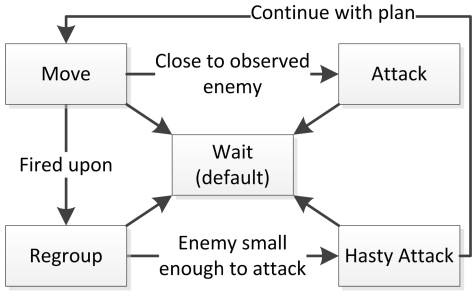

Figure 1. A context map defines all possible contexts and the transitions between them.

Action knowledge describes how an agent should behave in a context. If a part of the behavior is shared with other major contexts, this behavior should be expressed as a *minor context* which controls the agent for a short period of time. There can be unlimited levels of minor contexts, but one or none has generally been sufficient for most applications.

Knowledge of when to switch to another context composes the transition knowledge. This includes recognition of a situation leading to deactivation of the active context and activation of a more suited context. This knowledge is contained in transition rules, with criteria for when the agent makes the transitions defined in the context map, and should include transition to a default context when no other context is applicable.

Declarative knowledge includes other properties of a context, e.g. parameters and a list of possible minor contexts. In addition, all agents can access information from a *global fact base* whose information is visible to all agents, e.g., terrain and weather, as well as a *local fact base* that contain information only available to a particular agent, e.g., its damage status, fuel reserve, etc.

Various context-driven behavior representations have also been proposed by other researchers[24–26]. Turner for example, proposes a behavior representation paradigm with many similarities to CxBR[25]. Turner uses the term *context-mediated behavior* (CMB) to describe the use of contextual schemas that contain both descriptive knowledge about a particular context as well as knowledge about how to behave in that context. In contrast to CxBR, more than one schema can be active at the same time in CMB. Features of the current situation are used to find matching schemas, which are merged to create the context of the agent. The search for appropriate schemas are only done when the situation changes significantly.

3 CxBR for a Hierarchy of Command Agents

Former applications of CxBR have dealt with low level, executing agents, where it is relatively easy to determine what can be a context. Gonzalez, Stensrud and Barret used automobile driving as an example where two of the possible contexts were “suburban driving” and “freeway driving”[27]. The contexts for a command agent are not as straightforward because the agent’s actions are not actions per se but rather commands, reports and communications. Imagine a command agent having three subordinates. The

agent commands the first to scout for enemies, the second to move forward and the third to follow and support the second. What is the context of the command agent? At some point, one of the subordinates encounters an unexpected enemy that it will have to consider. Does the context of the command agent change or only the context of the subordinate? When should a command agent change context? At what point do the reports received from its subordinate units represent a context change for the superior agent? These must be properly reflected in its transition knowledge.

Gonzalez et al. applied CxBR to a management agent, in the form of a virtual project manager[28]. In this work, the CxBR agent managed a virtual construction project and the work of other agents, always being aware of the situation, the needs of the project, and how to address these needs. When the situation had grown to become over-constrained as a result of events, it knew to kick the problem upstairs, so to speak, by requesting relief from one or more of the constraints by its human superiors. As long as everything went according to plan, the agent was in a major context called "Normal". In addition, it had one major context handling design changes, "DesignChange", and one for handling other events connected to budget or schedule, "ExternalEvent". When it realized it would not be able to complete the mission within schedule or budget, it transitioned to major context "Impasse", where it notified its human superior. This construction management agent was a single command agent and not part of a hierarchy.

Our command agents must be able to handle many more situations than the project manager agent described by Gonzalez et. al. How they should react to different events depends on their task and available resources. We suggest representing military tasks, as expressed in an operation order, as mission contexts and use different "Normal" major contexts for different mission contexts. As explained in section 2, a mission context contains an objective and a plan for reaching it. We interpret a plan consisting of a sequence of one or more major contexts as representing "Normal". In addition, a mission context contains major contexts organized in a context map describing how to address different situations that might occur in order to return to the context "Normal".

For all mission contexts, we have defined a basic plan, consisting of a sequence of major contexts and a context map. However, the agent must adapt the basic plan to fit a specific situation. An example of adapting the mission context *Seize*1. is given in figures 2(a) and 2(b). Adapting a plan includes modifying transition rules for determining when to transition to the next major context in the plan and possibly removing from the plan major contexts representing sub-goals that have been already fulfilled or have become irrelevant.

In addition to adapting a mission context to the current situation by pruning its basic plan and modifying transition rules, we imagine the possibility for an agent to replace major contexts based on available resources. For example an agent might replace a general major context like *Coordinate Attack* with a more specialized version like *Coordinate Fix-Attack*2. as illustrated in figure 2(c).

The actual decision making knowledge of our command agents is represented in the major contexts. Each major context represents management of a tactical pattern that includes tasks to the subordinate units together with synchronization criteria in order to fulfill the purpose of that context. A major context may be simple, like a script, or it may involve complex decision making expressed in almost any computational paradigm, e.g. production rules, behavior trees, neural networks etc.

The actions of command agents involve communication with other agents, as illustrated in figure 3. These actions are to command the subordinates, to send reports to the superior, and to communicate with agents at the same level. The commands can be new mission contexts for the subordinates or coordination instructions, like changing formation or rules of engagement.

The command agents in the multi-agent system are ordered in an hierarchy equivalent to the military units to which they belong. For example, for a battalion the multi-agent system consists of one agent for the battalion, one for each company in the battalion and one for each platoon, as illustrated in figure 4. A direct mapping between the agents and the real chain of command makes the design clear and simple, and also makes the agent system and decision making easy to understand for military experts. In addition, it prepares the system to be used for other applications such as studies of communication in the hierarchy.

Our multi-agent system takes as input an operational order, including a synchronization matrix, as illustrated in table 1. A synchronization matrix represents a tactical pattern chosen by the human user issuing the order, and consists of tasks to military units as well as information about when tasks are to be started relative to each other[29].

The different tasks in the order are transformed into mission contexts and each task is sent to the appropriate agent. When a task is scheduled to start, the agent who has the corresponding mission context adapts the basic plan for how to carry out that mission context and enters the first major context in the plan. The major context contains action rules that decide what mission contexts to send to

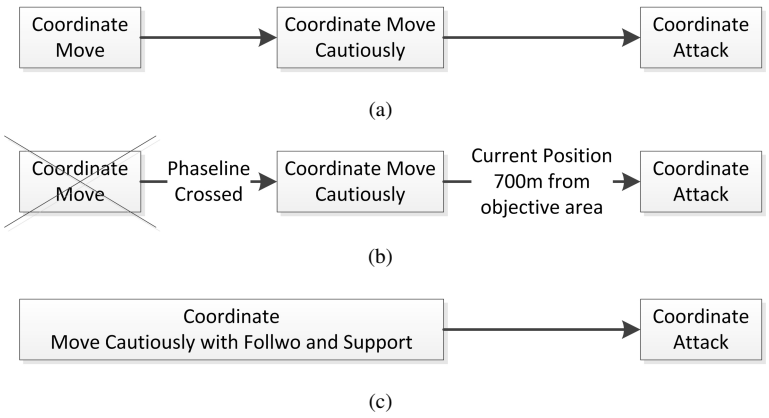

Figure 2. The figure illustrates an example of adapting the basic plan for mission context *Seize*(objective area, phaseline) to fit the current situation. (a) The basic plan of mission context *Seize*. (b) The plan for mission context *Seize* adapted to a situation when the phaseline is already crossed. Major context *Coordinate Move Cautiously* has been removed. (c) In the future we might want to specialize major contexts.

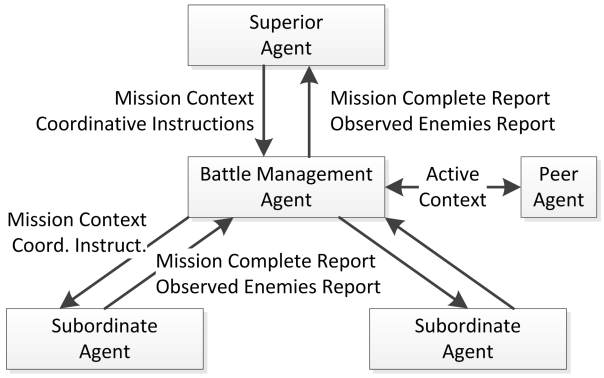

Figure 3. The figure illustrates what is communicated between a command agent (platoon or company) and its superior, subordinates and peers.

the subordinate agents. Equivalently, the subordinate agents complete their mission context plan, enter the first major context in the plan, and send mission contexts to their subordinates as specified by the action rules in the major context.

The lowest level agents represent leaders and staff of entities or aggregate units represented in the CGF system. The multi-agent system includes representation of these executing entities or aggragate units in addition to the command agents. These so called CGF-agents receive commands in the form of mission contexts as the other agents, and map these to low level commands that are sent to the CGF system. The CGF-agents are regularly updated with status information from the corresponding entities or aggregates in the CGF system.

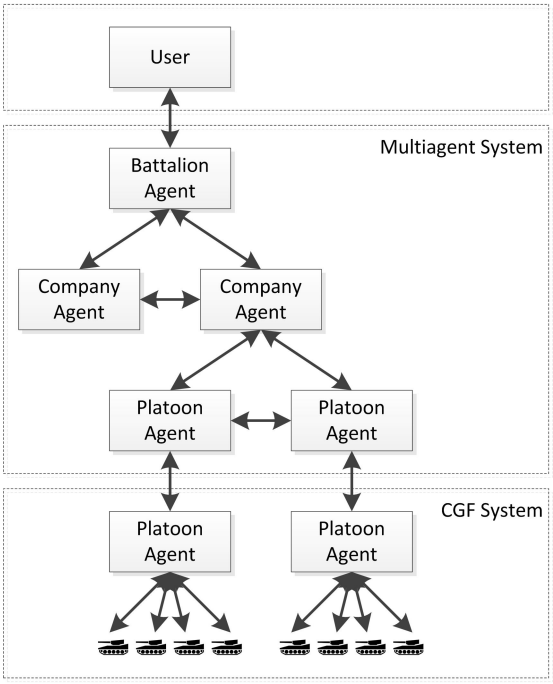

Figure 4. The multi-agent system is ordered in a hierarchy representing the chain of command. The multi-agent system receives orders from the user and command entities in a CGF system. The multi-agent system also receives reports from the executing agents and sends requests to the user.

The command agents make decisions based on reports they receive from their subordinates. What kind of information an agent is monitoring and how the information is used depend on its current context. For example, if the lowest level agent is in major context *Move* and receives reports on observed enemy presence, the agent will transition to

major context *Regroup*, where it will decide on how to handle the unexpected enemy. However, in major context *Attack* enemy presence is expected, and the agent will not change context if it receives information about observed enemies.

When an agent is not able to handle a situation it encounters, it will ask its superior for help, which results in a context transition for the superior. For example, if the lowest level agent that transitioned into the major context *Regroup* concludes that the observed enemy is too large for it to handle by itself, it will ask its superior agent to handle the problem. The superior agent will then transition to major context *Regroup* where it will deliberate on what to do next. Therefore, an agent will only transition to a major context that is not part of the "Normal" plan when a subordinate reports that it is not able to continue its mission because of the existence of some situation it cannot handle by itself.

4 Modeling an Example Scenario with Context-Based Reasoning

We have developed a simulation system consisting of a CxBR-based multi-agent system and the commercial CGF system, VR-Forces. This simulation system receives orders from a command and control information system (C2IS), as illustrated in figure 5. A C2IS is used by military officers for issuing commands to subordinate units and monitoring the progress of an operation. Ideally, commanding virtual forces should be carried out in the same way. The orders from the C2IS are high level orders[30–32], and the multi-agent system decomposes these orders into lower level tasks that are carried out by the entities in the CGF system.

In this section we provide an example of how we have used CxBR and a multi-agent system to represent and simulate battle management of a battalion operational order. We first describe the scenario and the input provided to the simulation system, before we present the CxBR models. In this section and the next, the command agents are sometimes called battalion, company and platoon agents.

4.1 Scenario

The scenario is the first part of a larger offensive operation. The goal of this operation is to destroy an enemy that is situated further to the north-east of the map shown in figure 6. The plan is to approach the enemy along the road that exits the map near the upper right corner. There is an enemy vanguard at area 102, and this area must be seized before the friendly forces can continue towards the main enemy position. The scenario considers this first part of the operation, namely to seize area 102.

Table 1. The synchronization matrix.

Company	Task		
Recce-Coy1	**Recon-noiter** axis from SL to 102	Keep 102 under surveillance	
MechInf-Coy2		**Seize** objective area 101	**Support by fire** MechInf-Coy3 towards 102
MechInf-Coy3			**Seize** objective area 102

This part of the operation is to be carried out by one battalion consisting of the companies RecceCoy1, MechInfCoy2 and MechInfCoy 3. The companies have one to three platoons, each of which consists of four vehicles. The battalion operation order is shown in table 1, which contains tasks for each company. In cooperation with subject matter experts, we determined how these tasks should be carried out and how the command agents should handle potential events not described in the order. We first outline the desired behavior we want the simulation system to display when everything proceeds as expected before describing how unexpected enemies are to be handled.

RecceCoy1 shall reconnoiter the axis from the start line SL to the objective area 102 while moving along the hillside to area 101. From the hill, the company will have a good overview of the road and the surrounding area. The company will search for enemy activities in the area and report to the other companies. RecceCoy1 will also survey the accessibility of the terrain, and the findings can be used by other companies for route planning. After bypassing area 101, it shall create an observation post at a suitable place for surveillance of area 102, where the vanguard is located.

MechInfCoy2 shall follow the same route as RecceCoy1, knowing that the route has been checked for enemies and obstacles. When MechInfCoy2 crosses phase line PL2, it shall prepare to attack potential enemies positioned in the objective area 101. After seizing area 101, it shall secure the area and move into position for overviewing the objective

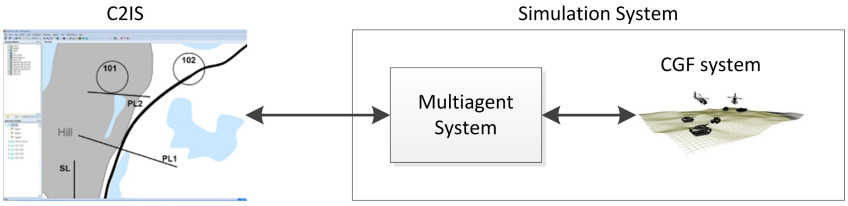

Figure 5. System Architecture

Figure 6. The map with defined areas and phase lines from the user defined battalion order

area 102, waiting for a signal to support MechInfCoy3 by fire during their attack on area 102.

MechInfCoy3 is to move along the road at high speed to phase line PL1, through the canalizing terrain along the lake. Upon reaching PL1, the company shall spread to a wider formation, preparing an attack on the vanguard in area 102. When the area is within weapon range, the attack is to begin, with support from MechInfCoy2. MechInfCoy3 should move into area 102 if they cannot engage the enemies from outside.

If unexpected enemies are detected, or if a vehicle is fired upon unexpectedly, the first reaction should be to withdraw and regroup. The next response will depend on the observation; small enemy unit, large enemy unit or an unidentified unit. If the enemy is considered sufficiently small compared to the platoon that has encountered it, the platoon should attack it and handle it alone. If the enemy is too large for the platoon to handle, but can be handled by the company, the rest of the company should assist in the attack. If the observed unit is unidentified, the observing entity should try to obtain more information about the unit.

4.2 Behavior Models

When the simulation system receives a operation order, it interprets the order and executes the battalion operation. In a simulation of the example scenario described above, the battalion agent in the multi-agent system receives the order in table 1 from the C2IS and makes sure the companies receive their tasks as scheduled in the order. The company agents plan their assigned tasks and command their platoons, and the platoon agents decompose their tasks into low-level commands for the platoon aggregates in the CGF system. Units below platoons (i.e. squads, vehicles, soldiers) are not represented in the multi-agent system, and the decomposition of platoon tasks into tasks for single entities is performed by the CGF system.

The use case operation order contains three tasks, which in the multi-agent system are represented by mission contexts: *Reconnoiter*, *Seize* and *Support by Fire*. These mission contexts are given to the company agents. For each mission context there is a context map containing the applicable contexts for a company agent performing that mission, and the possible transitions between these contexts.

Most of the mission contexts use the generic context map shown in figure 7. For mission context *X*, the plan only consists of one major context, *Coordinate-X*. The agent will therefore remain in this context as long as no unexpected enemies are encountered. The context *Coordinate-X* specifies how to decide which tasks are assigned to the subordinates, how to command them during the execution, and how to monitor the course of events. In addition to the major context *Coordinate-X*, which depends on the mission context *X*, the context map includes three major contexts that are identical for all mission contexts using this context map. The major contexts *Coordinate Regroup* and *Coordinate Hasty Attack* are used to handle unexpected enemies and the context *Coordinate Wait* is used when all planned contexts are completed or when waiting for commands or support from its superior. In our example, this generic context map is used by all

mission contexts, except for the company mission context *Seize*.

Mission context *Reconnoiter* uses the generic context map. A company agent receiving this mission will enter the major context *Coordinate Reconnoiter* and remain in this context throughout the execution of the task if no unexpected enemies are encountered. The mission context contains a route along which to reconnoiter, and the company agent commands its subordinate platoons to move cautiously along this route by assigning each their individual mission context *Move Cautiously*. The context map for this context is also the generic context map.

Mission context *Seize* uses the context map shown in figure 8. The plan of mission context *Seize* is a sequence of three major contexts: *Coordinate Move*, *Coordinate Move Cautiously* and *Coordinate Attack*. In addition to this plan, which is illustrated in figure 9, the context map contains the same contexts for handling unexpected enemy encounters as the generic context map. The mission context *Seize* contains a route to the objective area, and the company agent splits this route into three parts, one for each major context in the sequence.

In each of the major contexts *Coordinate Move*, *Coordinate Move Cautiously* and *Coordinate Attack*, the company agent sends its subordinate platoon agents the corresponding mission contexts *Move*, *Move Cautiously* and *Attack*, each with a route as a parameter. These mission contexts use the generic context map.

The generic context map is also used for mission context *Support by Fire*. The parameters of this mission context are the company to support and the objective area to attack. The company agent remains in the major context *Coordinate Support by Fire* if no unexpected enemies are encountered. The subordinate platoons are given the mission context *Move* to move into position to support by fire. The context map for *Move* is also the generic context map, and the platoons enter the context *Coordinate Move*. When the company agent receives a message that the company it is supporting has entered major context *Coordinate Attack*, it sends mission context *Attack* to its subordinates.

4.3 Implementation

The multi-agent system includes 1) classes for agents, mission contexts and major context, 2) a shared picture of observations in the CGF system, 3) internal services that are necessary to drive the internal functionality of the framework, like communication between agents and transition rule evaluation, and 4) external services that handle communication with the C2IS and the CGF system. Agent and context implementations will be explained further.

An agent has a mission context and is always in a major context. The agent behavior depends on the major context and is implemented there. A major context can use information from the mission context and other internal and external knowledge available to the agent. An agent has one superior agent and a list of subordinate agents, and it can send mission contexts to its subordinates and reports to its superior. When an agent is tasked to cooperate with other agents in a task, the mission context include references to the other agents witch will receive reports on the progression of the mission context.

A mission context is used to represent a military task expressed with the five Ws: “Who”, “What”, “Where”, “When”, and “Why”. “Who” specifies the tasked agent and possibly other agents that might be affected by this mission context. In the multi-agent system, the tasked agent is represented implicit by which agent has the mission context, and other agents are stored in two lists “supporting agents” and “affected agents”. “What” decides what type of mission context is instantiated, e.g. “SeizeMission”, “SupportbyFireMission”, etc. “Where” can be different for different types of mission contexts, e.g. a “SeizeMission” has a target area and possibly an operation area and an avenue of approach. “When” is represented by a start rule. It can be absolute (e.g. 12:00), relative to another task, or simply “start as soon as possible”. Future work might also include end time. “Why” is currently not supported. An example of a mission context implementation is illustrated in code example 1.

Code example 1 The pseudocode illustrates a simple version of mission context *Move*

```
▷ General major context parameters
private status
private startRule
private affectedWho[]
private supportingAgents[]

▷ Parameters specific for this mission context
private route
private targetLocation

procedure ISCOMPLETED(agent)
    return AtTargetLocation.isSatisfied(
                                agent.getLocation(),
                                targetLocation)
end procedure
```

A major context contains code for the actual decision making in a particular situation. The result is commands

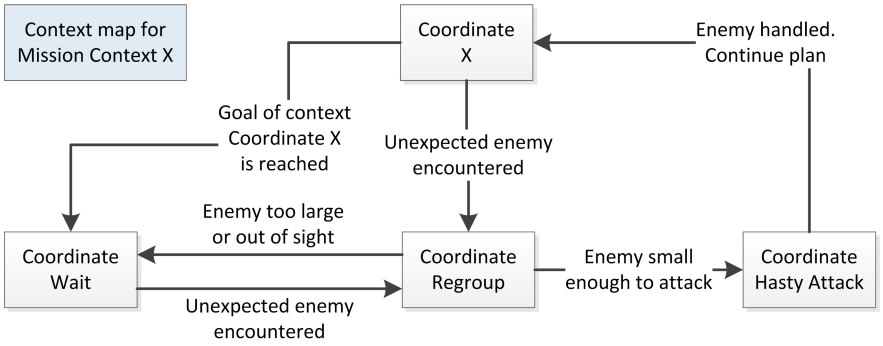

Figure 7. A generic context map used for most contexts.

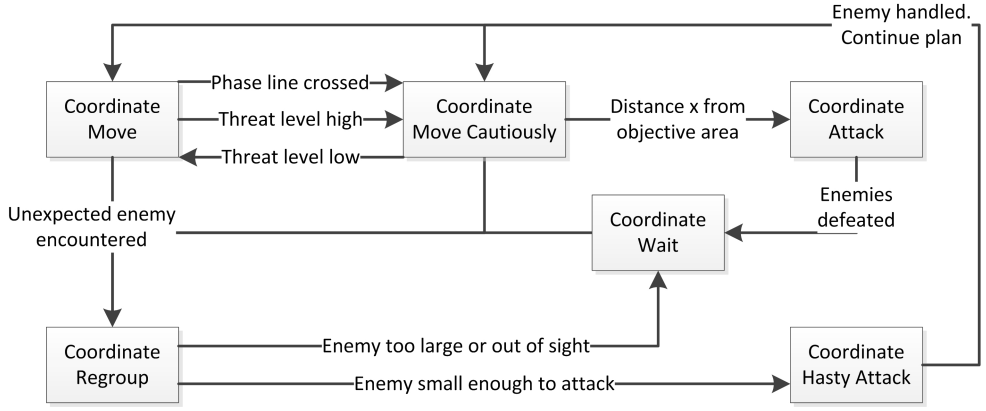

Figure 8. The context map for mission context *Seize*.

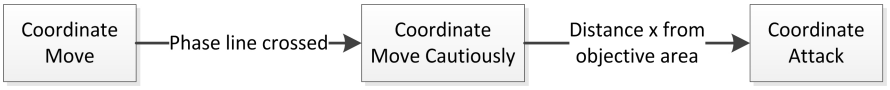

Figure 9. The context plan for mission context *Seize*.

to the subordinates, reports to the superior, and/or reports to peer agents. A major context must implement two methods, "init()" and "thick()", which include what to do when first entering the major context and what to do every timestep in that major context respectively. In the future we might add a method "end()" with what to do when leaving the context. Note that the decision making inside a major context might be represented by any behavior representation paradigm, e.g. neural networks, behavior trees, production rules etc. These can be mixed and matched to realize the desired behavior. A simple example of a major context implementation is illustrated in code example 2.

5 Tests and Evaluation

Several tests were run to evaluate the battle management model. The scenario and the behavior model described in section 4 are used in all these tests. In order to evaluate the behavior model, the strength, position and behavior of the enemy were varied. During testing the CxBR-based multi-agent system was used to control the friendly forces, whereas enemy forces were created and controlled by humans directly in the CGF system. The next section describes the tests in more detail, and the results are found in section 5.2.

5.1 Description of Tests

This section describes the tests for the evaluation of the planned and executed sequence of major contexts for one agent receiving mission context *Seize* (either MechInfCoy2 or MechInfCoy3) after the RecceCoy1 has reconnoitered the axis. Table 2 summarizes the conducted tests and their inputs.

Code example 2 The pseudocode illustrates a simple version of major context *Coordinate Regroup*.

```
▷ General major context parameters
private status

▷ Parameters specific for this major context
private routeAwayFromEnemy
private threatDistance
private isRetreating

procedure INIT(agent)
    SENDMISSIONTOALLSUBORDINATES(Wait)
    Retrieve enemyPosition from global fact base
    Retrieve agentPosition from local fact base

    routeAwayFromEnemy
        = CALCULATERUTEAWAYFROMENEMY(
                agentPosition, enemyPosition)
end procedure

procedure TICK(agent)
    Retrieve list of enemies inside threatDistance
    from global fact base
    if number of enemies > 0 then
        if enemy is too strong to handle then
            SENDREPORTTOSUPERIOR(
                new ThreateningEnemiesReport)
            SENDMISSIONTOALLSUBORDINATES(
                Move(routeAwayFromEnemy))
            isRetreating = true
        else
            status = Completed
            ▷ Can transit to HastyAttack
        end if
    else if isRetreating then
        ▷ Have retreated far enough
        SENDMISSIONTOALLSUBORDINATES(Wait)
    end if
end procedure

function FINDROUTEAWAYFROMENEMY(
                agentPosition, enemyPosition)
    ...
    return routeAwayFromEnemy
end function

procedure SENDMISSIONTOALLSUBORDINATES(
                mission)
    ...
end procedure

procedure SENDREPORTTOSUPERIOR(report)
    ...
end procedure
```

The input parameters of the tests are chosen to evaluate the ability of the command agents to assign tasks to their subordinates based on the conditions on the battlefield. The conditions are limited to the perceived presence, strength and location of the enemy. These input parameters are *enemy strength*, *enemy location*, and *knowledge of enemy presence*.

The parameter *enemy strength* represents the relative strength of the enemy, if any, compared to the friendly unit tasked with mission context *Seize*. The possible values are *weak*, *strong*, and *no enemy*. A weak enemy can be handled by the tasked unit, while a strong enemy would require support from another unit.

Enemy location indicates whether the enemy, if any, is located within or outside an objective area (OA). The parameter values are *inside OA*, *outside OA*, and *no enemy*.

The *knowledge of enemy presence* reflects the observations reported by the reconnaissance company RecceCoy1. The value *yes* indicates that the enemy has been observed. This implies that its strength has been assessed. The value *no* indicates that no enemy has been observed.

5.2 Results

The output of the tests consists of the context plan as well as the actual sequence of major contexts during the execution of the mission. The results of the tests are presented in an abbreviated form, according to the abbreviations in table 3.

In table 4 we compare the agent's planned and executed sequence of major contexts with what we should expect based on the CxBR model implementation. The objective is to verify that we have implemented what we designed. As we see from the table there are no differences between the actual sequences of major contexts and the expected ones. This indicates that the model was implemented correctly.

In table 5 we validate the results by comparing them to what the major contexts in the plan and during execution should have been. In this table we note some discrepancies in tests number 3, 5, 6, and 9. We discuss these in the next section.

5.3 Discussion of results

For the most part the resulting behavior was as desired, however, the validation revealed three weak points in the model. First, observed enemies outside of objective areas are not considered when adapting the basic plan. Second,

Table 2. Test inputs.

Test Id	Enemy Strength	Enemy Location	Enemy Known
1	no	no	no
2	weak	outside OA	no
3	weak	outside OA	yes
4	strong	outside OA	no
5	strong	outside OA	yes
6	weak	inside OA	no
7	weak	inside OA	yes
8	strong	inside OA	no
9	strong	inside OA	yes

Table 3. Abbreviations

Abbrevation	Major Context
M	Coordinate Move
MC	Coordinate Move Cautiously
A	Coordinate Attack
HA	Coordinate Hasty Attack
R	Coordinate Regroup
W	Coordinate Wait

the strength of the enemy is not evaluated when the enemy is located in the objective area. Third, the plan may become invalid as new information is gathered during execution. These three issues will be discussed next.

By comparing the results of test 2 and 3, and also 4 and 5, we see that the information gathered by the reconnaissance company about enemies outside the objective area is never used. This is seen because the plan is the same whether enemies have been observed or not. These observations should be taken into account when planning the operation.

The second issue concerning the lack of consideration of the strength of the enemy inside the objective area is revealed as the results are identical for the tests with a weak enemy inside the objective area (6 and 7) and those with a strong enemy within objective area (8 and 9). The enemy strength should be taken into account during planning and execution.

The need for replanning is made apparent by tests 6 and 8. Since there are no known enemies inside the objective area when adapting the basic plan, the major context *Attack* is removed from the plan. This plan is not changed as it should be when enemies inside the objective area are observed later during execution.

6 Discussion

In our experience, an important advantage of using CxBR was that the behavior model became well structured. By organizing the behavioral knowledge into different contexts, CxBR helped to keep the behavior model clear and easy to explain to non-experts. Furthermore, it appears relatively straight forward to expand the model, and we expect this will make it easy to implement a user interface that illustrates and explains the decisions the agents make.

For both company and platoon agents, the mission and major contexts were tightly connected to the behavior of the corresponding unit. For example, when a company was tasked to reconnoiter, the company's command agent received the mission context *Reconnoiter* and its decision making was encapsulated in the major context *Coordinate Reconnoiter*. We argue that this is sensible because the decisions made by the commander are heavily dependent on the military task. Additionally, it made the behavior model easy to relate to for subject matter experts, which is beneficial for validation and makes the model more credible for intended users.

Researchers at TNO developed a similar system as ours consisting of a multi-agent system that simulated battle management and controlled the same CGF system VR-Forces. Different from us they used JADEX, a Belief-Desire-Intention (BDI) reasoning engine for intelligent agents[33]. Our two efforts were compared in a joint paper[34]. The conclusion was that both CxBR and BDI were suitable for developing command agents, but had different pros and cons. The benefit of using BDI-agents is that BDI is a well-known paradigm for agent modeling, with several available frameworks, which makes it easy to start with. CxBR on the other hand is more modular, which might make it easier to understand the whole behavioral model and avoid inconsistencies as the behavior model grows larger.

The Command Forces (CFOR) simulation is another effort on developing command agents that model the capabilities of a human military commander[35;36]. The command agents were developed using the SOAR cognitive architecture[37] and the behavior revolves around making plans, monitor the execution of the plan and replan when the plan is no longer valid. The planner has the ability to maintain multiple plans and reason about their interactions. The agent use this to try to predict the behavior of other peer agents as well as the opposing force and how this might affect its own plan.

Mason and Moffat have work on representing command and control decison-making in CGF systems to support operational analysis studies[38;39]. They also focused on planning, and divided the planning into two levels. The command agents at the top level implement a deliberate planning strategy that search for the optimal allocation of resources. At the lower level the command agents use

Table 4. Verification of the CxBR model.

Test Id	Result Plan	Result Execution	Expected Plan	Expected Execution
1	M-MC	M-MC	M-MC	M-MC
2	M-MC	M-R-HA-M-MC	M-MC	M-R-HA-M-MC
3	M-MC	M-R-HA-M-MC	M-MC	M-R-HA-M-MC
4	M-MC	M-R-W	M-MC	M-R-W
5	M-MC	M-R-W	M-MC	M-R-W
6	M-MC	M-MC-R-HA-MC	M-MC	M-MC-R-HA-MC
7	M-MC-A	M-MC-A	M-MC-A	M-MC-A
8	M-MC	M-MC-R-W	M-MC	M-MC-R-W
9	M-MC-A	M-MC-A	M-MC-A	M-MC-A

Table 5. Validation of the CxBR model.

Test Id	Result Plan	Result Execution	Desired Plan	Desired Execution
1	M-MC	M-MC	M-MC	M-MC
2	M-MC	M-R-HA-M-MC	M-MC	M-R-HA-M-MC
3	M-MC	M-R-HA-M-MC	M-MC-A-M-MC	M-MC-A-M-MC
4	M-MC	M-R-W	M-MC	M-R-W
5	M-MC	M-R-W	W	W
6	M-MC	M-MC-R-HA-MC	M-MC	M-MC-A
7	M-MC-A	M-MC-A	M-MC-A	M-MC-A
8	M-MC	M-MC-R-W	M-MC	M-MC-R-W
9	M-MC-A	M-MC-A	W	W

pattern matching to match the current situation to a situation stored in memory. Each stored situation is linked directly to a course of action appropriate to that situation, where a course of action is represented as a synchronization matrix. The plan execution is monitored at both levels, and replanning is conducted when needed.

Our work differs from these other approaches in that immediate threats and possibly opportunities are handled without replanning. In our system a threat will trigger a change in major contexts, but when the threat is handled, the agent can continue the execution of the plan were it left of. The same mechanisms can be used to satisfy implicit goals like maintaining supplies, security and communication. We believe this is a major improvement to pure planning systems.

In our work planning has not yet been a major focus. In CxBR a plan is a part of the mission context and consists of a sequence of major contexts. Little research has been done on how to compose this plan. Typically, this plan is predefined for each specific mission context. In our model we adapted a basic, predefined plan to the actual situation. This made the plan fit the situation more effectively than it would without this adaption. However, the results summarized in section 5 show that replanning is still required as the situation changes. We see the simplified planning and the lack of replanning as major weaknesses with the current model, and we aim to improve this in future work.

7 Conclusion and Future Work

Context-based reasoning has previously been used mostly for single agents or two cooperating agents and for executing agents. In this paper, we describe an application of CxBR to a hierarchy of command agents. The agents in the hierarchy represent the commanders and their staff in the military forces. The hierarchy consists of one battalion agent, company agents and platoon agents. These agents receive tasks and divide them into more detailed tasks, which they give their subordinates. They also receive reports on detected enemies and how a battle is proceeding from their subordinates. The subordinates of the platoon agents are platoons in a CGF system, which execute the tasks in the virtual battlefield.

We modeled a small selection of tasks with CxBR in order to study whether CxBR is suited for modeling the behavior of command agents. The result was well structured behavior models that were comprehensible for military officers. This was an advantage both in the behavior modeling process and when demonstrating the resulting models, as the modeled behavior was easy to explain and discuss with subject matter experts.

The work in this paper is a first attempt on using CxBR for a hierarchy of command agents. We have considered cooperation and communication between several agents. We have developed behavior models for a small set of tasks in order to explore the suitability of CxBR for this purpose and to identify subjects for future work. We have not focused on variations of how each task should be executed, and the behavior models are in some respects fairly simple. However, our conclusion is that the CxBR paradigm seems suitable for modeling battle management as well as lower level behavior, and we plan to improve our behavior models in future work. Possible subjects for future work are more advanced behavior in each context, several variations of behavior and planning for each task depending on the situation, and more advanced situation assessment.

Author Biographies

Rikke Amilde Løvlid is a Scientist at FFI – Norwegian Defence Research Establishment, Kjeller, Norway where she works with artificial intelligence in both simulation- and robotic systems. She received a PhD in computer science from the Norwegian University of Science and Technology (NTNU) in 2013.

Solveig Bruvoll is a Senior Scientist at FFI – Norwegian Defence Research Establishment, Kjeller, Norway and an associate professor at the University of Oslo, Section for Autonomous Systems and Sensor Technologies.

Karsten Brathen is a Chief Scientist at FFI – Norwegian Defence Research Establishment, Kjeller, Norway and a Research Manager for the Modeling and Simulation research program at the Cyber Systems and Electronic Warfare Division, FFI. He has more than 15 years of experience in management, research and development of defense modeling and simulation.

Avelino Gonzalez is a Professor in the Computer Science Department at the University of Central Florida in Orlando, Florida. His area of expertise is Artificial Intelligence, with special focus on context driven behavior representation. He is currently investigating learning human tactical behavior from unintrusive observation of human performance in a simulator.

Notes

1. "Seize" is "a tactical mission task that involves taking possession of a designated area by using overwhelming force"[40].

2. "Fix" is "a tactical mission task where a commander prevents the enemy from moving any part of his force from a specific location for a specific period"[40].

References

1. Builder CH, Bankes SC and Nordin R. *Command Concepts: A Theory Derived From the Practice of Command and Control*. RAND Corporation, 1999.
2. Gonzalez AJ. Tactical reasoning through context-based reasoning. In Brézillon P and Gonzalez JA (eds.) *Context in Computing: A Cross-Disciplinary Approach for Modeling the Real World*. New York, NY: Springer New York, 2014. pp. 491–508.
3. Gonzalez FG, Grejs P and Gonzalez AJ. Autonomous automobile behavior through context-based reasoning. In *Proceedings of the International Florida Artificial intelligent Research Society Conference*. 2000.
4. Brown JC. *Application and Evaluation of the Context-based Reasoning Paradigm*. Master's Thesis, Department of Electrical and Computer Engineering, University of Central Florida, Orlando, 1994.
5. Grejs PF. *Autonomous Automobile Behavior through a Context-based Approach*. Master's Thesis, Department of Electrical and Computer Engineering, University of Central Florida, Orlando, 1998.
6. Patz BJ, Papelis Y, Pillat R et al. A practical approach to robotic design for the DARPA urban challenge. *Journal of Field Robotics* 2008; 25: 528–566.
7. Gonzalez AJ and Ahlers RH. Concise representation of autonomous intelligent platforms in a simulation through the use of scripts. In *Proceedings of the Sixth Annual FLAIRS Conference*. 1993.
8. Gonzalez AJ and Ahlers R. Context-based representation of intelligent behavior in training simulations. *Transactions of the Society for Computer Simulation International* 1998; 15(4): 153–166.
9. Gumus I and Gonzalez AJ. A threat ranking algorithm for multiple intelligent entities in a simulated environment. In *Proceedings of the Florida Artificial Intelligence Research Society Conference*. 1999.
10. Proenza R. *A Framework for Multiple Agents and Memory Recall within the Context-based Paradigm*. Master's Thesis, Department of Electrical and Computer Engineering, University of Central Florida, Orlando, 1997. Master's thesis.
11. Hu Zt and Liu J. The research of ASW helicopter ACGF construction based on CxBR. In *Proceedings of the International Conference on Computational Intelligence and Security Workshops*. 2007.
12. Fazenda P, Carreira P and Lima P. Context-based reasoning in smart buildings. In *Proceedings of the First International Workshop on Information Technology for Energy Applications*, volume 923. 2012. pp. 855–874.
13. Norlander L. *A Framework for Efficient Implementation of Context-based Reasoning in Intelligent Simulations*. Master's Thesis, Department of Electrical and Computer Engineering, University of Central Florida, Orlando, 1999.

14. Devero L and Gonzalez AJ. Collaborative teamwork representation in context-based reasoning. In *Proceedings of the 2001 Computer Generated Forces and Behavior Representation Conference*. 2001.
15. Barrett GA and Gonzalez AJ. Effective agent collaboration through improved communication by means of contextual reasoning. *International Journal of Intelligent Systems* 2011; 26(2): 129–157.
16. Grama C, Gonzalez AJ, Pollak E et al. Automated plan generation for mid-echelons in the military decision-making process. In *Proceedings of the Spring Simulation Interoperability Workshop*. 1998.
17. Fernlund HK, Gonzalez AJ, Georgiopoulos M et al. Learning tactical human behavior through observation of human performance. *IEEE Transactions on System, Man, and Cybernetics - part B: Cybernetics* 2006; 36(1): 128–40.
18. Stensrud BS and Gonzalez AJ. Discovery of high-level behavior from observation of human performance in a strategic game. *IEEE Transactions on System, Man, and Cybernetics - part B: Cybernetics* 2008; 38: 855–874.
19. Trinh VC and Gonzalez AJ. Discovering contexts from observed human performance. *IEEE Transactions on human-machine systems* 2013; 43: 359–370.
20. Johnson CL and Gonzalez AJ. Learning collaborative team behavior from observation. *Expert Systems with Applications* 2014; 41(5): 2316–2328.
21. Stein G and Gonzalez AJ. Learning in context: enhancing machine learning with context-based reasoning. *Applied Intelligence* 2014; 41(3): 709–724.
22. Wooldridge M. *An Introduction to MultiAgent Systems*. John Wiley & Sons Inc., 2002.
23. Russel S and Norvig P. *Artificial Intelligence - a modern approach*. 3 ed. Prentice Hall, 2010.
24. Brézillon P and Pomerol JC. Context in artificial intelligence: I. A survey of the literature. *Computer & Artificial Intelligence* 1999; 18(5): 1–34.
25. Turner RM. Context-mediated behavior for intelligent agents. *International Journal of Human-Computer Studies* 1998; 48: 307–330.
26. Brézillon P, Pasquier L and Pomerol JC. Reasoning with contextual graphs. *European Journal of Operational Research* 2002; 136(2): 290 – 298.
27. Gonzalez AJ, Stensrud BS and Barret G. Formalizing context-based reasoning: A modeling paradigm for representing tactical human behavior. *International Journal of Intelligent Systems* 2008; 23: 822–847.
28. Gonzalez AJ, Tsuruta S, Sakurai Y et al. Using contexts to supervise a collaborative process. *International Journal of Artificial Intelligence for Engineering Design, Analysis and Manufacturing (AIEDAM)* 2010; 25(1): 25–40.
29. Headquarters Department of the Army. Field manual no. 5-0, The Operations Process, 2010. URL `fas.org/irp/doddir/army/fm5-0.pdf`.
30. Carey SA, Kleiner MS, Hieb M et al. Standardizing battle management language - a vital move towards the army transformation. In *Proceedings of the 2001 Fall Simulation Interoperability Workshop*. 01F-SIW-067, 2001.
31. NATO. Modelling and Simulation Group 085: Standardization for C2-simulation interoperation, NMSG-085 final report. Technical report, 2014.
32. Hyndoy JI, Mevassvik OM and Brathen K. Simulation in support of course of action development in operations. In *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference (I/ITSEC)*. 2014. pp. 1844–1854.
33. Pokahr A, Braubach L and Lamersdorf W. Jadex: a BDI reasoning engine. In *Multi-Agent Programming - Languages, Platforms and Applications*, *Multiagent Systems, Artificial Societies, and Simulated Organizations*, volume 15, chapter 6. Springer, 2005. pp. 149–174.
34. Løvlid RA, Alstad A, Mevassvik OM et al. Two approaches to developing a multi-agent system for battle command simulation. In *Proceedings of the 2013 Winter Simulation Conference (WSC 2013)*. Washington, DC, 2013. pp. 1491–1502.
35. Salisbury MR, Brooker LB, Seidel DW et al. Implementation of command forces (CFOR) simulation. In *Proceedings of the 5th Workshop on Computer Generated Forces and Behavior Representation*. Florida, FL, 1995.
36. Hill RW, Gratch J and Rosenbloom PS. Flexible group behavior: Lessons learned building virtual commanders. In *Proceedings of the 9th Conference on Computer Generated Forces and Behavior Representation*. Florida, FL, 2000.
37. Laird JE. *The Soar Cognitive Architecture*. The MIT Press, 2012.
38. Mason CR and Moffat J. An agent architecture for implementing command and control in military simulations. In *Proceedings of the 2001 Winter Simulation Conference (WSC 2001)*. 2001. pp. 721–729.
39. Mason CR and Moffat J. Representing the C2 process in simulations: Modelling the human decison-maker. In *Proceedings of the 2000 Winter Simulation Conference (WSC 2000)*. 2000. pp. 940–949.
40. Headquarters Department of the Army . Field manual no. 3-90, Tactics, 2001. URL `www.globalsecurity.org/military/library/policy/army/fm/3-90/`.