Simulation, Visualization, and Validation of Adaptive Multi-Agent Organizations in Naval Applications

Mark Hoogendoorn[1], Catholijn M. Jonker[2], Martijn C. Schut[1], and Jan Treur[1]

[1]Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{mhoogen, schut, treur}@cs.vu.nl
[2]Radboud University Nijmegen, Nijmegen Institute for Cognition and Information,
Montessorilaan 3, 6525 HR Nijmegen, The Netherlands
C.Jonker@nici.ru.nl

Abstract

The naval domain is characterized by a dynamic environment. This requires constant adaptation of the organisation, choosing between a wide variety of options. The consequences of the different options are difficult to foresee and hence, it is hard to judge which option is best. This paper presents automated support for the simulation, visualization, and validation of such adaptive multi-agent organisations. Generic simulation properties are specified using a formal modelling approach. Furthermore, results of a realistic case study are presented, and validated by means of properties obtained from naval experts. Finally, a tool is introduced that enables an insightful visualization of the simulation results.

INTRODUCTION

Agent-based modeling techniques are often used to model and simulate (natural or artificial) agent systems that have to deal with dynamic and uncertain environments. Therefore, an important challenge for the area of agent-based modeling is the notion of adaptivity. Adaptation can take place within a single agent (e.g., an individual learning process), or at the level of the multi-agent organization (e.g., change of roles of agents within the organization). In order to create (multi-)agent-based simulations with adaptive abilities, adaptation mechanisms have to be incorporated in agent-based simulation models.

Adaptation mechanisms can involve not only quantitative numerical aspects but also qualitative, logical aspects (for example, a role switch between agents within an organization). If formalization is used for an adaptation mechanism, this is often based on mathematical models using differential equations. In contrast, agent-based simulation models traditionally make use of qualitative, logical languages. Most of these languages are appropriate for expressing qualitative relations, but less suitable to work with more complex numerical structures as, for example, in differential equations. Therefore, integrating such mathematical models within the design of (multi-)agent-based simulation models is difficult. To achieve this integration, it is needed to bridge the gap between quantitative approaches and the type of languages typically used in agent-based simulation.

In the area of simulation, a formalized model is used to compute the simulation steps [9]. Languages and software environments are available to support this modeling process. Validation of a model is usually not formally supported; it is considered a different issue. Often validation is done informally, by hand (or eye), based on comparison of a simulation trace with an empirical trace. In addition, sometimes specific (e.g., statistical) techniques are used to support certain aspects of validation. Usually in the domain that is modeled, *global properties* that should hold for the behavior of a simulation model can be identified. As the languages used to specify a simulation model are directed to *local properties* (the steps between successive states), such global properties cannot be formalized in these languages. To obtain more support, also for validation of a simulation model, it is needed to integrate the modeling of such global properties in a formal manner as well, so that their specification and automated checking on simulation traces also can be supported by the modeling environment.

In accordance with the findings mentioned above, this paper introduces an approach for simulation and analysis of adaptive (multi-)agent systems and underlying mechanisms that is integrative in two ways:

(1) It combines in one modeling framework both *qualitative, logical* and *quantitative, numerical* aspects
(2) It allows to model dynamics at *different aggregation levels*, from a more local level (e.g., behaviors of roles within the organization) to a global level (behavior of the multi-agent organization as a whole); moreover, *interlevel relations* can be specified that express relationships between dynamic properties at different levels

Modeling dynamics at a local level often concerns expressing temporal relationships between pairs of successive states, such as described, for example, by basic steps within an adaptation mechanism. Local level specifications are the basis for the computation steps for a simulation model. From the more global perspective, more complex relationships over time can be used to model dynamics for adaptive multi-agent organizations. For example, how during a history of events to which it adapts, the system's behavior is changing.

Interlevel relations often take the hierarchical form of an AND-tree (or a number of them), with the most global property at the top (root) and the most local at the leaves. Such a hierarchical structure can be useful in the analysis of, in case, why a global property fails on a certain simulation trace. By going down in the tree and at each level checking the properties under the failing node, finally the leaf or leaves that fail(s) can be found, thus pinpointing the (local) cause of the failure. This can be useful in debugging a model, but also in the analysis of the circumstances under which a model will function well and under which not, and the reasons why.

The main objective of the research described in this paper is to investigate the suitability of a system involving planning, simulation, visualisation, and validation with respect to automated planning support in naval missions. The longer term aim of this research is to contribute to the development of a tool that allows for personnel to plan with a confidence and speed that would not be otherwise possible.

The remainder of this paper is structured as follows. Section 2 gives some details about the naval domain addressed and how adaptive organisation forms play a role. In Section 3, the modelling methodology that has been used is presented. Section 4 presents a number of simulations that have been conducted and describes a case study that has been investigated. Section 5 presents the plan visualisation tool. Section 6 describes validation results for the case study. Finally, Section 7 concludes and describes future work.

ADAPTIVE ORGANIZATION WITHIN THE NAVAL DOMAIN

Within the dynamic naval environment actions of possibly opposing parties, but also possible interference of non-military bystanders might induce a need for change in the organisation to ensure the safety of the mission. Which response to choose in a given situation depends on a variety of factors. Elements such as enemy resources and innocent bystanders have to be taken into consideration and it is hard to predict the consequences of a plan that has been chosen. This paper presents an automated support system for the simulation, visualization, and validation of such processes. Two requirements must be met concerning such support: 1) the support must agree with the current way of working, and 2) guarantees must be given over the resulted planning with respect to given conditions including intended outcome and required resources. The work presented here researches an approach for implementing automated support that meets these two requirements.

As the current way of working is concerned, the naval domain knows a large volume of well thought out plans that are scheduled for and during a mission (the so-called 'doctrine'). Everyone involved in a mission is familiar with these plans. The performed planning during a mission consists mostly of switching between and carrying out those plans. On the one hand, such planning during a mission may be a matter of executing the plans that were decided upon for the mission; on the other hand, unexpected events may happen that ask for necessary replanning during a mission. Concerning the latter, these situations require appropriate and speedy response. It is essential that in these situations, current circumstances are taken into account, a suitable plan is selected from the doctrine, the situation is dealt with and the mission will continue as originally planned.

Replanning in the naval domain frequently involves organisational change: it actually affects the organisational structure in addition to adaptation of other plans. For example, in response to an unexpected event, a ship that was originally only an escort of a high-value unit, may have to change its role to an attack unit. Such replanning situations are not rare: organisational changes are frequent and substantial.

Another important aspect of naval planning involves spatial information. Feasibility of a plan is partly determined by the nature of the available resources (helicopters, frigates, transporters) and the relative location of those resources. Combining the specific capabilities of the resources with spatial information and timing aspects plays a key role in the planning. Therefore plan visualisation that includes spatial information is necessary for successful implementation of automated planning support in naval applications.

In naval missions, it is crucial to consider the planning within the broader context of mission goals, available resources, intended outcomes, etcetera. In this respect, performed planning before and during a mission must be checked against such kinds of conditions. For example, when an agent is reallocated to another role (e.g., because of prevailing circumstances), it must inform others at the time that it is able to fulfill its role. It is important to recognise that this reallocation does not happen instantaneously (e.g., because a ship may have to sail towards some location to fulfill its new role), and therefore the communication is essential for others to know when the agent can receive orders in its new role.

This paper presents a simulation model that includes: a planner (P) for organisational change; a simulator (S) for those plans that reflects the meta-knowledge (see for example [5]) of the roles involved regarding organisational change; a visualisation tool (VS) for the spatial effects of plan execution that is dedicated to the naval domain; and a validation tool (VL) for the validation of the resulting planning.

The essential virtue of the model is that it recognises the importance of spatial information in naval planning (by means of the visualisation) and it offers an inventive way to check whether given conditions hold while planning (by means of the validation). The model may be used offline for analysis purposes and/or mission planning, as well as during execution of a mission as an automated planning support tool.

MODELLING METHODOLOGY

To facilitate formal modelling of a multi-agent organisation and its dynamics, this section introduces an organisational modelling approach and, in addition, a modelling language that enables specifying the dynamics within an organisation (see also [3]). The organisational modelling approach is described in Section 3.1, and the formal language for expressing dynamics is addressed in Section 3.2.

AGR Organisation Modelling Approach

For the description of actual multi-agent organisations, the AGR (for agent/group/role) model has been adopted [2]. In that approach, an organization is viewed as a framework for activity and interaction through the definition of groups, roles and their relationships. But, by avoiding an agent-oriented viewpoint, an organization is regarded as a structural relationship between a collection of agents. Thus, an organization can be described solely on the basis of its structure, i.e. by the way groups and roles are arranged to form a whole, without being concerned with the way agents actually behave, and multi-agent systems will be analyzed from the outside, as a set of interaction modes. The specific architecture of agents is purposely not addressed in the organizational model. The three primitive definitions are:

- The *agents*. The model places no constraints on the internal architecture of agents. An agent is only specified as an active communicating entity which plays roles within groups. This agent definition is intentionally general to allow agent designers to adopt the most accurate definition of agent-hood relative to their application. In this paper, the agents are however assumed to be reflective agents, allowing them to reason about the role they are playing.

- *Groups* are defined as atomic sets of agent aggregation. Each agent is part of one or more groups. In its most basic form, the group is only a way to tag a set of agents. An agent can be a member of n groups at the same time. A major point of these groups is that they can freely overlap.

- A *role* is an abstract representation of an agent function, service or identification within a group. Each agent can handle multiple roles, and each role handled by an agent is local to a group. Roles can also have beliefs due to the assumed reflective capabilities of the agents; they can reason about whether they should have a particular belief given a certain role. These beliefs can be seen as an additional requirement on the agents playing that role.

Modelling Language for Dynamics

In this section a method to express dynamics within an organisational model is addressed. To formally specify dynamic properties that are essential in an organisation, an expressive language is needed. To this end the Temporal Trace Language is used as a tool; cf. [7]. For the properties occurring in the paper informal, semi-formal or formal representations are given. The formal representations are based on the Temporal Trace Language (TTL), which is briefly defined as follows.

A state ontology is a specification (in order-sorted logic) of a vocabulary. A state for ontology Ont is an assignment of truth-values {true, false} to the set At(Ont) of ground atoms expressed in terms of Ont. The *set of all possible states* for state ontology Ont is denoted by STATES(Ont). The set of *state properties* STATPROP(Ont) for state ontology Ont is the set of all propositions over ground atoms from At(Ont). A fixed *time frame* T is assumed which is linearly ordered. A *trace* or *trajectory* γ over a state ontology Ont and time frame T is a mapping $\gamma : T \rightarrow$ STATES(Ont), i.e., a sequence of states γ_t ($t \in T$) in STATES(Ont). The set of all traces over state ontology Ont is denoted by TRACES(Ont). Depending on the application, the time frame T may be dense (e.g., the real numbers), or discrete (e.g., the set of integers or natural numbers or a finite initial segment of the natural numbers), or any other form, as long as it has a linear ordering. The set of *dynamic properties* DYNPROP(Ont) is the set of temporal statements that can be formulated with respect to traces based on the state ontology Ont in the following manner.

Given a trace γ over state ontology Ont, the input state of some role r within a group g at time point t is denoted by

state(γ, t, input(r|g))

analogously

state(γ, t, output(r|g))
state(γ, t, internal(r|g))

denote the output state and internal state.

These states can be related to state properties via the formally defined satisfaction relation |=, comparable to the Holds-predicate in the Situation Calculus: state(γ, t, output(r|g)) |= p denotes that state property p holds in trace γ at time t in the output state of role r within group g. Based on these statements, dynamic properties can be formulated in a formal manner in a sorted first-order predicate logic with sorts TIME or T for time points, Traces for traces and F for state formulae, using quantifiers over time and the usual first-order logical connectives such as $\neg, \wedge, \vee, \Rightarrow, \forall, \exists$. In trace descriptions, notations such as

state(γ, t, output(r|g))|= p

are shortened to

output(r|g)|p

To model direct temporal dependencies between two state properties, the simpler *leads to* format is used. This is an executable format defined as follows. Let α and β be state properties of the form 'conjunction of literals' (where a literal is an atom or the negation of an atom), and e, f, g, h non-negative real numbers. In the *leads to* language $\alpha \twoheadrightarrow_{e, f, g, h} \beta$, means:

If state property α holds for a certain time interval with duration g,

then after some delay (between e and f) state property β will hold for a certain time interval of length h.

For a precise definition of the *leads to* format in terms of the language TTL, see [8]. A specification of dynamic properties in *leads to* format has as advantages that it is executable and that it can often easily be depicted graphically.

SIMULATIONS

This Section presents the simulator component within the system. First of all, the executable (leads to) properties which specify the simulation model for the simulator are presented in Section 4.1. After that, Section 4.2 addresses the case study that has been investigated, followed by the results of the simulations of the case study.

Simulation Model Specification

This Section describes generic properties that constitute the basis for the simulation model. Each of these generic properties can be formed into more scenario specific rules whenever necessary. The generic properties in the framework work are based on goals, plans, beliefs and events. The used architecture is shown in Figure 1.
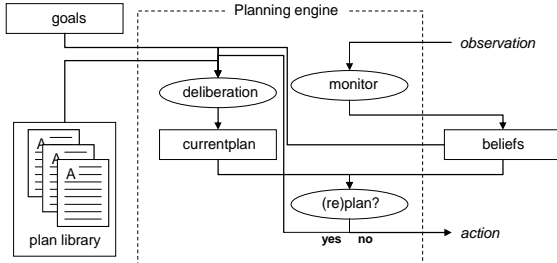
Figure 1. Architecture of the simulation model

It has to be mentioned that beliefs in this respect are used for storing information about the environment as well as information about oneself. As shown in the scenario below, many plans involve *organisational change*. This means that the actual organisational structure changes as the result of some occurring event. Thus, in addition to knowing about the environment by observation, it is assumed that the agent (reflectively) knows about its role in the organisation and can change to another role if necessary. The architecture shown in Figure 1 has been formalised and this formalisation is explained in the remainder of this section. Firstly, it is assumed that a goal has been given.

internal(r:ROLE|gr:GROUP)|belief(g:GOAL, pos)

denotes that role *r* within group *gr* holds the belief that *g* is a goal. Based on this goal, a plan is selected to achieve it:

internal(r:ROLE:gr:GROUP)|belief(current_plan(p:PLAN), pos)

says that plan *p* is selected as to achieve goal *g*. This plan will generate actions as long as no disturbing events occur. If such an event occurs and *r* is informed, this is denoted by

input(r:ROLE:gr:GROUP)|communication_from_to(r1:ROLE|gr1: GROUP, r:ROLE|gr:GROUP ,inform, e:EVENT)

stating that *r1* within group *gr1* informs *r* within group *gr* about event *e*. This event causes another goal to become active.

internal(r:ROLE|gr:GROUP)|belief(g1:GOAL, pos)

says that *g1* is now a goal and a subsequent plan is selected:

internal(r:ROLE|gr:GROUP)|belief(current_plan(p1:PLAN), pos)

This plan may involve organisational change. If this is the case (as it is in the scenarios below), a modelling approach is adopted as developed elsewhere [6]. This involves the existence of a ChangeManager who directs the organisational change. This approach is explained in more detail below. If the plan has been fully executed, this is denoted by

internal(r:ROLE|gr:GROUP)|belief(plan_executed(p:PLAN),pos)

where the parameter might be left out if it is assumed that only one plan can be executed at a time. This causes role *r* to reflect on other still existing goals and resuming the plans to achieve these goals. If there are no existing goals, a new goal may be generated or given.

Execution of a certain plan that has been selected often consists of organisational change. Therefore, generic simulation rules for these organisation structure changes are needed to enable a generic simulation model. The properties shown below are based on the approach presented in [6] which is partially based on the AGR organisation modelling approach as presented in Section 3.1. In that approach, organisational change can be performed in a meta-group called ChangeGroup, in which Member roles are present that represent agents within the organisation. Each agent in the organisation is represented by exactly one Member role within the ChangeGroup. The Member roles have beliefs about the organisation and these beliefs are transferred to the roles the agent is currently playing. To initiate the change process as described above, triggers are needed. These are specified in the current plan, and are domain specific. Given this specific information for the particular plan, generic simulation rules fire to simulate the process of informing the members involved and changing their current beliefs on the organisation. Some example rules are presented below.

RP(ChangeManager):Communicate Activity
[output(ChangeManager|ChangeGroup)|communication_from_to(
        ChangeManager|ChangeGroup, all_involved, inform,
        active(C:CHANGE_GROUP)) &
internal(ChangeManager|ChangeGroup)|belief(involved_in_group(
        R:ROLE, C:CHANGE_GROUP), pos)]
$\rightarrow_{0,0,1,1}$
input(R:ROLE|ChangeGroup)|communication_from_to(
        ChangeManager|ChangeGroup, R:ROLE|ChangeGroup,
        inform, active(C:CHANGE_GROUP)

RP(Member): Believe Change Activity
input(R:ROLE|ChangeGroup)|communication_from_to(
        ChangeManager|ChangeGroup, R:ROLE|ChangeGroup,
        inform, active(C:CHANGE_GROUP)
$\rightarrow_{0,0,1,1}$
[internal(R:ROLE|ChangeGroup)|belief(active(C:CHANGE_GROUP,
        pos) &
 output(R:ROLE|ChangeGroup)|communication_from_to(
        R:ROLE|ChangeGroup, ChangeManager|ChangeGroup,
        inform, belief(active(C:CHANGE_GROUP), pos))]

Rules such as the examples above cause the ChangeGroup to be activated, knowledge about a new structure to be communicated, and finally belief emerging at the roles that need to have this information. After all of this has been performed, the ChangeGroup is deactivated and the new structure is in place (part of the internals of the roles).

Roles are attributed with reflective knowledge in the approach presented in this paper. This means that roles have beliefs on the expected behaviour concerning the role. For example, a role has the internal belief that when the role receives an input *x* he eventually has to output *y*, formally:

internal(Role|Group)|belief(leadsto(input(Role|Group)|*x*,
output(Role|Group)|y, efgh(0,0,1,1)),pos)

Simulation Results

This section contains results of simulations using the model presented in Section 3 and the generic properties presented in Section 4.1 which have been formalised in terms of the formal languages presented in Section 3. First of all, the case study is introduced, thereafter some example formal properties which

specify the behaviour in the situations that occur in the case study are shown. Finally, the simulation trace for the case study is shown.

Case study

This section presents a case study that has been obtained from experts of the Royal Netherlands Navy. The scenario contain events that are typical within the naval domain. The scenario that has been studied is called *submarine threat*. The initial configuration of the fleet is shown in Figure 2. In total there are six frigates, denoted by F1 - F6, each allocated to a certain area within which they reside. Besides the frigates there are also helicopters (H1- H6) flying in a particular zone of the fleet. Finally, there are certain High Value Units (HVU) within the area called ZZ (for Zulu Zulu) that need to be protected. These might for example be ships containing troops, or amphibian landing vehicles. In total there are five ships within ZZ, which is called MainBody throughout this paper. At a certain point in time the Officer in Tactical Command (OTC) receives an assignment to sail to Peterselie island and chooses a fleet configuration. On the way however, after a certain time-point however, frigate F1 detects sonar contact with a high probability that it is a submarine. The OTC now has to plan the actions to be performed to deal with such an event.

Case Specific Formal Properties

This section presents some example properties that have been formalized to enable the simulation of the submarine threat case study, if a role informs the OTC that sonar contact with a submarine has been made, he forms a search and attack unit:

RP(OTC): Handle sonar contact
input(OTC|Fleet)|communication_from_to(R:ROLE|Screen1,
        OTC|Fleet, inform, sonarcontact_sub)
$\rightarrow_{0,0,1,1}$
internal(OTC|Fleet)|belief(current_plan(eliminate_submarine_threat),
        pos)

The plan to eliminate such a submarine threat involves forming a search and attack unit. In case such a unit if formed, a new group is created called SAU. Furthermore, the role of commander within the SAU, the SAUC is performed by the agent previously allocated to LeftProtector1. Formal:

RP(OTC): Perform plan to form SAU
∀A:AGENT, R:ROLE, G:GROUP
[internal(ChangeManager|ChangeGroup)|current_plan(
        eliminate_submarine_threat), pos) &
 internal(ChangeManager|ChangeGroup)|belief(allocated_to(
        A:AGENT, LeftProtector1, G:GROUP), pos)]
$\rightarrow_{0,0,1,1}$
internal(ChangeManager|ChangeGroup)|belief(delete(allocated_to(
        A:AGENT, FrontLeftProtector1, G:GROUP)), pos) &
internal(ChangeManager|ChangeGroup)|belief(add(exists_group(
        SAU)), pos) &
internal(ChangeManager|ChangeGroup)|belief(add(exists_role(
        SAUC)), pos) &
internal(ChangeManager|ChangeGroup)|belief(add(allocated_to(
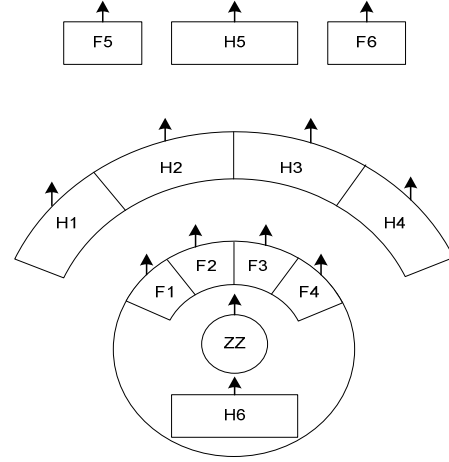   A:AGENT, SAUC, SAU)), pos)]

Figure 2. Initial Fleet configuration

Formal Simulation Trace

The results of the case study that has been performed are presented here. Figure 3 shows the trace regarding the simulation of the "submarine threat" case study. Briefly, the trace shows the following elements: First of all, OTC is informed by the LeftProtector1 within Screen1 about a sonar contact with a sub. At that same time-point the OTC derives a new plan:

internal(OTC|Fleet)|belief(current_plan(
eliminate_submarine_threat, pos)

As a result, a search and attack unit (SAU) is formed again, and the submarine is located. After the location is known, the OTC orders the rest of the fleet to turn away. The command is confirmed by the ships within the MainBody1 and they eventually communicate to have turned away:

input(OTC|Fleet)|communication_from_to(
BodyMember1|MainBody1, OTC|Fleet, inform, turned_away)

Following the observation that the ships must be outside of range for the torpedo's, the ships are told to turn back to their old direction again. All confirm and execute the order. The OTC commands the helicopters to replace the frigates that take part in the SAU because the helicopters are much faster and the distance between the SAU and the rest of the Fleet is increasing.

output(OTC|Fleet)|communication_from_to(OTC|Fleet,
LeftDetector|Screen1, inform, replace_sau)

Due to the open position in Screen1 that is left, helicopter F3 is allocated to two roles within the Screen. After a certain time, the OTC believes the submarine in no threat anymore and orders the roles within the SAU group to return to their mother ship:

output(OTC|Fleet)|communication_from_to(OTC|Fleet,
SAUC|SAU, inform, return_to_mothership)

This denotes that at a later point in time, the helicopter is allocated to the role of FrontLeftProtector1 within Screen1, which is already allocated to frigate F2:

Figure 3. Simulation result of Submarine Threat Scenario

internal(OTC|Fleet)|belief(allocated_to(H1, FronLeftProtector, Screen1)

The commands to refuel and change the crew of the helicopter are therefore sent to the role to which F1 and H2 are allocated. After the refuel is done, the old fleet configuration is restored.

VISUALIZATION

For the simulator a visualization tool has also been developed. Figure 4 shows a screenshot of the tool. On the left side of the figure the fleet is shown in a visual manner as previously shown in Figure 3 whereas on the right side the trace (of which parts were explained already in Section 4.2), that acts as a basis for the visualization, is shown. A bar in the trace shows the accompanying time-point for which this visualization holds. For Navy domain experts such a visualization tool is easily interpretable whereas a trace as shown on the right side of Figure 4 is hard to interpret especially due to the fact that one needs to be familiar with such kind of formalisms.

VALIDATION

When a formalized trace has been obtained either by a formalization of an empirical trace or by means of simulation it is useful to verify certain essential properties in the trace. Below the properties that have been checked against the trace presented in Section 4 are shown. The properties are independent from the specific scenario and should hold for every trace. The properties are formalized using Temporal Trace Language as described in Section 3.

P1: Reflective Behaviour

This property states that in case a role has a belief about an executable property that should be fulfilled when the role is being performed, the role should actually show this behaviour. Formally:

$\forall\gamma$:TRACES, t1:TIME, A:ANTECEDENT, C:CONSEQUENT
$[\exists$ R:ROLE, G:GROUP
state(γ, t1, internal(R|G)) |= belief(leadsto(A, C, efgh(_,_,_,_)), pos)
$\Rightarrow [\forall$t2 $\geq$ t1 state(γ, t2) |= A $\Rightarrow \exists$t3 $\geq$ t2 state(γ, t3) |= C]]

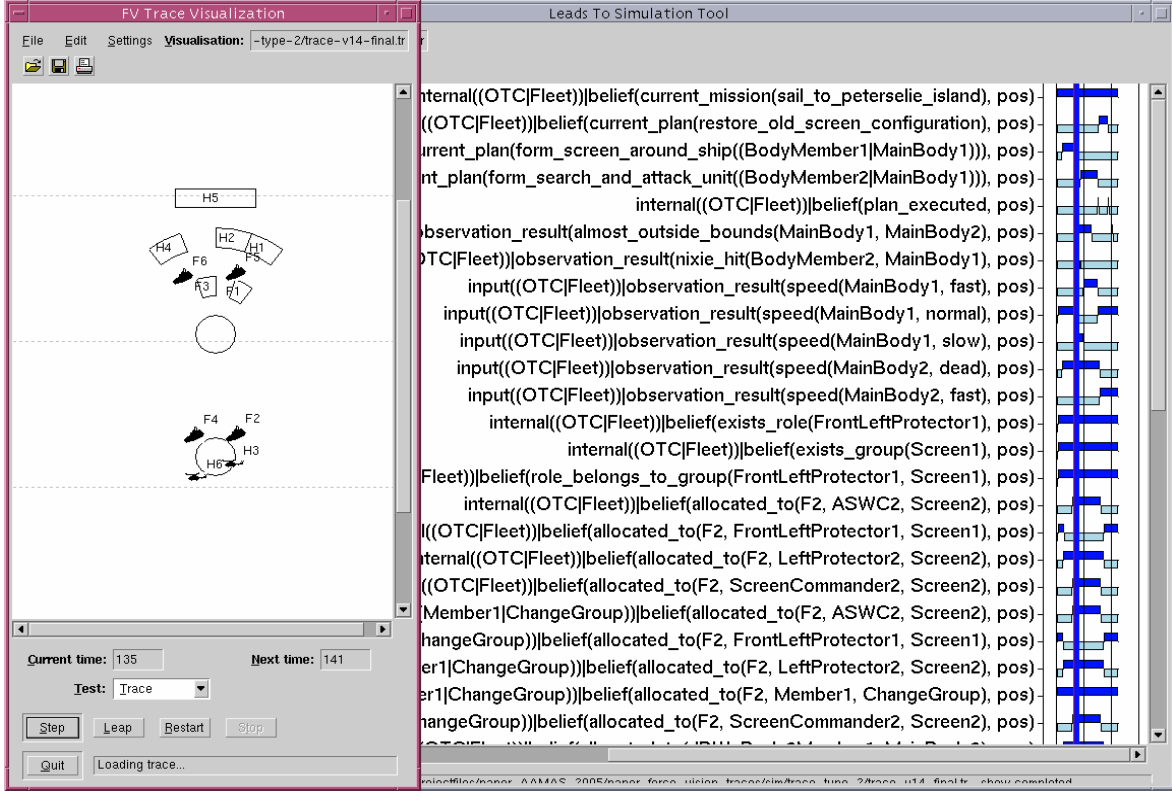This property is indeed satisfied for the presented traces.

Figure 4. Screenshot of the visualization tool

P2: Ship always allocated to a role

The fact that a ship should always be allocated to a role (after the initial fleet setup) is specified using this property. In formal form the property is formulated as follows:

∀γ:TRACES, t:TIME > 20, A:AGENT
  [∃R:ROLE, G:GROUP
   state(γ, t, internal(OTC|Fleet)) |= belief(allocated_to(A, R, G), pos)]

This property is also satisfied for the given traces.

P3: Communication that an agent is able to fulfil its role

This property entails that when an agent is re-allocated to another role, it should always communicate when it is able to fulfil the role. There can be a time-delay between the re-allocation because the ship might have to sail to a particular place to execute the newly assigned role. Formally the property can be specified in the following way:

∀γ:TRACES, t1:TIME > 20, A:AGENT, R:ROLE, G:GROUP
 [∃R2:ROLE state(γ, t1, input(ChangeManager|ChangeGroup)) |=
                 communication_from_to(R2|ChangeGroup,
                 ChangeManager|ChangeGroup, inform,
                 belief(add(allocated_to(A, R, G)), pos))
   ⇒ [∃t2:TIME ≥ t1 state(γ, t2, output(R|G)) |=
          communication_from_to(R|G, OTC|Fleet, inform,
                      able_to_fulfil_fole)]]

This property is satisfied as well for the given traces.

P4: Determine a plan to handle exceptions

When an exception occurs the OTC within the fleet always has a belief about a current plan that handles the exception:

∀γ:TRACES, t1:TIME
  [∃E:EXCEPTION state(γ, t1, input(OTC|Fleet)) |= E ⇒
   ∃t2:TIME ≥ t1, P:PLAN [state(γ, t2, internal(OTC|Fleet)) |=
belief(current_plan(P), pos)]]

This property is satisfied for the trace presented in Section 4.

DISCUSSION

This paper introduces an integrative modeling approach for simulation and analysis of adaptive behavior of multi-agent organizations. The approach is integrative in two ways. First, it combines both *qualitative, logical* and *quantitative, numerical* aspects in one modeling framework. Second, it allows to model dynamics at *different aggregation levels* from local to more global levels.

The organizational processes during naval missions have been formalized by identifying executable local dynamic properties for the basic dynamics. On the basis of these local properties simulations have been made. Moreover, dynamic properties describing the behavior at a global level have been identified. These properties have been checked automatically on the simulation traces. To this end a system has been introduced that consists of

four components: (1) A planning component; (2) a simulation engine; (3) a visualization tool, and (4) a component which enables formal validation. The planning component has been equiped with typical plans for the naval domain from the so called 'doctrine'. The simulation engine has as a basis an organizational model which is specified by means of dynamics in the form of formal executable properties. Organizational change and change of plans are visualized in an understandable manner for naval experts by means of the visualization tool. Finally, the validation component enables formal validation of traces.

The approach taken in this paper has a number of advantages over other approaches. When comparing with planning achicitures such as [4] and [1], the approach presented in this paper provides validation functionalities for the simulation results, which is not the case in the other architectures. The models of these architectures can be formally proven to be correct, however for the complex naval domain it might be too diffult to prove such a thing. Furthemore the approach in this paper also has the ability to validate and visualize empirical traces who can for example be obtained from logbooks. These advantages could be used to monitor a current mission, and constantly check whether the properties that should hold for the mission are satisfied. In case a property is not satisfied, a warning could for example be given.

Other simulation engines have been developed specifically for the naval domain, such as for example presented in [10]. For a matter of validation of the model however, navy experts were asked what they considered to be the optimal solution. In the approach used in this paper, this process is automated due to the formal specification of properties provided to us by naval domain experts.

ACKNOWLEDGEMENTS

REFERENCES

1. d'Inverno, M., Luck, M. Georgeff, M., Kinny, D. and Wooldridge, M., The dMARS Architechure: A Specification of the Distributed Multi-Agent Reasoning System. Journal of Autonomous Agents and Multi-Agent Systems, 9(1-2):5-53, 2004.
2. Ferber, J. and Gutknecht, O., A meta-model for the analysis and design of organisations in multi-agent systems. In: Proceedings of the Third International Conference on Multi-Agent Systems (ICMAS'98), IEEE Computer Society Press, pp. 128-135.
3. Ferber, J., Gutknecht, O., Jonker, C.M., Müller, J.P., and Treur, J., Organization Models and Behavioural Requirements Specification for Multi-Agent Systems. In: Y. Demazeau, F. Garijo (eds.), Multi-Agent System Organisations. Proc. of the 10th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'01.
4. Georgeff, M. P., and Ingrand, F. F., Decision-making in an embedded reasoning system. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89), pages 972-978, Detroit, MI, 1989.
5. Goodwin, R., Meta-Level Control for Decision-Theoretic Planners. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1996.
6. Hoogendoorn, M., Jonker, C.M., Schut, M., and Treur, J., Modelling the Organisation of Organisational Change. In: Proc. of the Sixth International Workshop on Agent-Oriented Information Systems, AOIS'04.
7. Jonker, C.M., Treur, J. Compositional verification of multi-agent systems: a formal analysis of pro-activeness and reactiveness. International. Journal of Cooperative Information Systems, vol. 11, 2002, pp. 51-92.
8. Jonker, C.M., Treur, J., and Wijngaards, W.C.A., A Temporal Modelling Environment for Internally Grounded Beliefs, Desires and Intentions. Cognitive Systems Research Journal, vol. 4, 2003, pp. 191-210.
9. Law A.M. and Kelton D.W., Simulation, Modeling and Analysis. McGraw Hill, 2000. Third edition.
10. Sokolowski, J., Enhanced Military Decision Modeling Using a MultiAgent System Approach, In Proceedings of the Twelfth Conference on Behavior Representation in Modeling and Simulation, Scottsdale, AZ., May 12-15, 2003, pp. 179-186.