Game Environment for Command and Control Operations (GECCO)
General Description

Joel Brynielsson, Henrik Bäärnhielm, Andreas Enblom, Jing Fu Zi, Niklas Hallenfur, Karl Hasselström, Henrik Hägerström, Oskar Linde, Klas Wallenius, and Jon Åslund

Department of Numerical Analysis and Computer Science
Royal Institute of Technology
SE-100 44 Stockholm
Sweden
`gecco@nada.kth.se`

14th May 2001

Contents

1 Introduction

Computer simulations of real world phenomena gains importance in all sciences. As computer power increases the real world simulations get more detailed. We will study schematic simulated worlds called *microworlds*[3]. In such microworlds it is possible to provide the players with the latest technology available today as well as the technology of tomorrow.

Many examples of realistic microworlds can be seen in the war games available for personal computers on the commercial market. The games available today are, however, not suited for research purposes[13]. The main reasons for this are that you are not allowed to register information and that you do not have access to the source code in order to make changes “on the fly”.

In GECCO we have implemented a game specifically for the research community. GECCO is a strategy game where you move units on a map. The game is generic in the sense that it stores all information regarding the scenario in files that are easy to adjust. The source code is distributed to the research community as “open source”. The source code is well documented and well structured. The game works in all computer environments[1].

2 Game characteristics

2.1 Geographical environment

The geographical environment in GECCO consists of an automaton matrix. The automaton matrix is automatically constructed from an image which is read by the server. Each pixel in the image is classified as a predefined automaton (i.e, forest, buildings, road, water etc.). The classification is performed by looking at the pixel color and then looking up this color in a configuration file for the particular game. If there is no exact matching with a color, as is often the case when you for example use a map with several green colors representing the forest, the color with the shortest euclidean RGB distance is chosen.

Different automatons may have different properties, for example forest automatons might start to burn, water automatons may get polluted etc.

2.2 Units

Units are vector-based in order to make it possible to calculate what positions in the automaton matrix that the unit affects. A unit holds an arbitrary number of properties that describes the state of the unit. Such properties may be fuel, health, ammunition, water, food, etc.

A client may have *observe* or *command* rights on a unit. If a client *observes* a unit, the client receives all information that the unit has in its posession. If a client *commands* a unit, the client is also able to send *actions* (see section 2.3) to the unit. Typically a player has got command rights on his own units, observe rights on his friends units and no rights at all on units that belongs to his enemy.

[1]GECCO is implemented in Java and requires a set of computers connected via an ordinary TCP/IP-network.

2.3 Actions

Actions are used for interaction between automatons and units. When a unit or an automaton wants to do something, like puting another automaton on fire, attacking another unit or distinguish fire, it sends an action to the unit or automaton that it wants to affect. Specific actions are specified for specific automaton and specific units. An automaton may affect another automaton, a unit may affect another unit, a unit may affect an automaton and an automaton may affect a unit.

3 GECCO as a tool for command and control

To be able to use GECCO for command and control, we provide functionality for *ghost units*. A ghost unit is simply an icon that the user can move around on the map in the way that he or she prefers. The ghost unit has nothing to do with the game simulation inside the server. It is used solely for planning purposes and gives the player an oppurtunity to place enemy units on areas where the player thinks the enemy units currently are present. Primarily we think of situations where an enemy unit has disappeared from the players field of vision, and the player wants to predict where this unit is heading.

The concept of ghost units resides completely inside the client. The icon that represents a ghost unit is the icon that represents the original unit, but it is significantly shaded in grey color. A ghost unit possess no intelligence at all. It is solely an icon that the player controls.

A ghost unit may occur in two different ways:

- A ghost unit is automaticly presented on the map when a real unit disappears from the field of vision. The icon is chosen according to the icon that represented the real unit. This functionality may be turned off via a menu, which may be good in situations with many fast moving units that would otherwise make the map crowded with ghost units.
- A player can create a ghost unit by himself by clicking on the right mouse button on a spot on the map, where the ghost unit is then located. When the button is clicked the player may choose unit from a menu containing all units that are defined for the particular game.

4 User characteristics

We categorize GECCO users into three categories:

Player The player is a person that takes part in a game. He play a role in a scenario that simulates the microworld that he is to practice within. The player does not need to have any knowledge about GECCO or the computer environment that is used for GECCO.

User The user is a person that uses predefined scenarios in order to manage a GECCO session. The user needs to have basic knowledge of how to start a GECCO session and also needs to know how the computer network that he is running GECCO on works. The user is able to make small changes in the scenarios regarding quantities, amount of players, permissions for different players etc. The user uses the *GECCO User's Manual*[6].

Scenario creator The scenario creator is a person who creates new scenarios. This person needs to have brief knowledge on how Java works. The scenario creator uses the *GECCO Developer's Manual*[5].

5 Implementation

The game code is divided into three different parts, *server*, *client* and *communication*. Each part is strictly separated from the other parts using object oriented programming techniques. In order to be able to run GECCO in all possible computer environments, all code is written in Java. All code is well documented using Javadoc.

5.1 Server

The server is the engine of the game. It holds the game simulation and distributes the information to the clients. One GECCO session requires one server.

As much of the functionality as possible has been put in the server. The map, for example, is being read by the server and then distributed to the clients.

5.2 Client

A GECCO session can take any number of clients, depending on the scenario and the requirements. A client may be a human that maneuver his units using a graphical user interface representation or a client may be automated by a computer.

The client does not possess any knowledge about the scenario. All definitions regarding a particular scenario (i.e., map data, unit data, data regarding the troops etc.) are stored at the server side and transferred to the client on startup.

At all times a client "see" different parts of the map, depending on where the units that the client control and observe are located. If a client see a certain spot on the map, this spot is shown using clear colors while the points that the client does not see are shown in shaded grey "ghost" colors. The server gives the client information about changes in the map, as well as information regarding what spots on the map that the client is currently able to see.

A unit may be *active* in a certain client. This means that the unit is shown on this clients map and that the client takes part in information sent from the server regarding this unit. If the client has got control or observe rights on this unit (i.e., a "friend" unit), this means that the client gets information regarding all properties that concerns the unit (e.g., fuel health etc.). If the client has not got control or observe rights (i.e., an "enemy" unit), the client only receives information regarding the position.

As of today we have implemented a non-automatic client that gives a human player the possibility to command units. This client holds a graphical user interface displaying all units that the player controls along with units that he knows of. This client also provides insight in what areas you see at the moment, as well as it distinguishes real units from ghost units.

To be able to make complex maneuvers our client gives the player the possibility to “queue” his forthcoming actions. Using this technique, the player can tell one of his units to make multiple movements in order to move along a road or to move around a lake.

5.3 Communication

The communication part of the implementation defines a protocol that completely separates the clients from the server.

The protocol takes care of the initial handshaking process when objects are transferred from server to client. When started the following messages may be sent from the server to a client:

1. an automaton changes its state,
2. a unit becomes active, i.e., it becomes “visible”,
3. a property within an active unit changes state,
4. changes in the field of vision, i.e., an automaton or unit disappears from the field of vision,
5. reply to confirm action request from a client (see below).

The only possible message that a client may send to the server is the following:

1. action request, i.e., a request from a unit to perform an action (see section 2.3) of some kind (attack a unit, distinguish fire etc.).

6 Example scenarios

As a start we have developed three test scenarios. The scenarios have been chosen for two reasons:

1. they are different from each other in the sense that they will make use of the architecture in different ways and therefore evaluates the GECCO implementation from many different perspectives,
2. they are all examples of practical implementations of scenarios used in current command and control research at the Swedish National Defence College[21].

During the development phase we have also developed a test scenario, much alike the war games on the commercial market, that is quite entertaining to play.

6.1 World War 2

This scenario represents a large scale operation in Europe at the operative level. The map represents a large area. The time perspective concerns days rather than hours. We think that this scenario would fit nicely for training of military commanders in command and control environments such as ROLF[15, 16, 17].

In an architectural point of view this scenario is interesting because the units will interact with eachother in many complex ways.

6.2 The Öresund-bridge collapses

Among the 12 scenarios defined in [21], this is perhaps the most heavily discussed one, see for example [2]. A tanker hits the Öresund-bridge, train wagons and cars fall into the freezing water, oil is leaking from the tanker threatening to pollute the coast of southern Sweden, oil is burning, etc.

This scenario requires helicopters and boats to interact with the catastrophe area. From an architectural point of view this means that we will have a lot of interesting interaction between units and automatons.

6.3 Fire fighting

Fire fighting is a heavily studied topic at the Swedish National Defence College who are running simulations with a decision support system specifically designed for forest fire fighting called C^{3}FIRE[10, 11] on a regular basis.

The fire fighting scenario developed for GECCO is based on the same ideas as C^{3}FIRE and makes use of the same algorithms as C^{3}FIRE when simulating the fire development. A number of new units have been added, such as a "bandit" car that ignites fires and fire extuinguishing helicopters.

6.4 "The commercial war game" – a tactical scenario

When developing the game we needed a scenario to test our ideas on. We used a scenario that looks like a simple version of the war games available today. The scenario uses a map with roads, lakes etc. covering a small part of Sweden. There are three types of units in the scenario, tanks, helicopters and radar stations, all having its own properties (i.e., the helicopters can travel across lakes, the tanks can take a lot of damage, the radar stations has got a broad field of vision etc.).

7 Future plans

7.1 Research

The primary purpose of GECCO is to supply the research community with an interesting research environment. Primarily we think of research within the following areas:

- development of tools for decision support that can be connected to manual clients in the game, see for example [7],
- development of computer generated forces, CGF:s, i.e., "smart" clients that are controlled by the computer. Many interesting and stimulating techniques are available for this task[14], and there exist several interesting projects at present[9, 22],
- development of tools for statistical treatment of logfiles.

7.2 Command and control training

Decision making in emergency- and military organizations is classified as *distributed decision making*[4] which means that the decision making is distributed among the actors in the organization. These ideas influence the ongoing research in command and control[1, 8, 12, 15, 16, 20], and also seem to be the goal for products currently under development[18, 19].

In the world of today there are many emergency organizations that rely on *operation management* from their staff. The situations that occur are different from time to time and do often behave in a very complex manner.

Another organization that relies on the operation management of the staff is the military. Here there are command and control centers that have to give the right orders.

A common problem for these organizations is that they need to practice. In most of the situations (e.g., a forest fire or a war) it is very expensive and inconvenient to actually let the staff practice in a real world scenario. A (partial) solution to this problem is to let the personnel practice in microworlds that simulate reality.

Within the Swedish National Defence College, command and control simulations are run within the ROLF project[15, 16, 17]. We think that GECCO will be of value for the researchers within the ROLF project.

7.3 Open source project

The intention is to provide GECCO as open source to the research community. It will be distributed using GNU Public License (GPL), meaning all code is free for everyone to use.

The GECCO project was once initiated as a cooperation between researchers and students. This way of creating software for research purposes has shown to be a viable way of working. Hopefully there will come oppurtunities to engage students in future GECCO open source development. The process of developing GECCO is continous and therefore the academic world is well suited for this process.

There are plenty of ideas on the wish lists that hopefully will come true in the proposed open source project. Some examples:

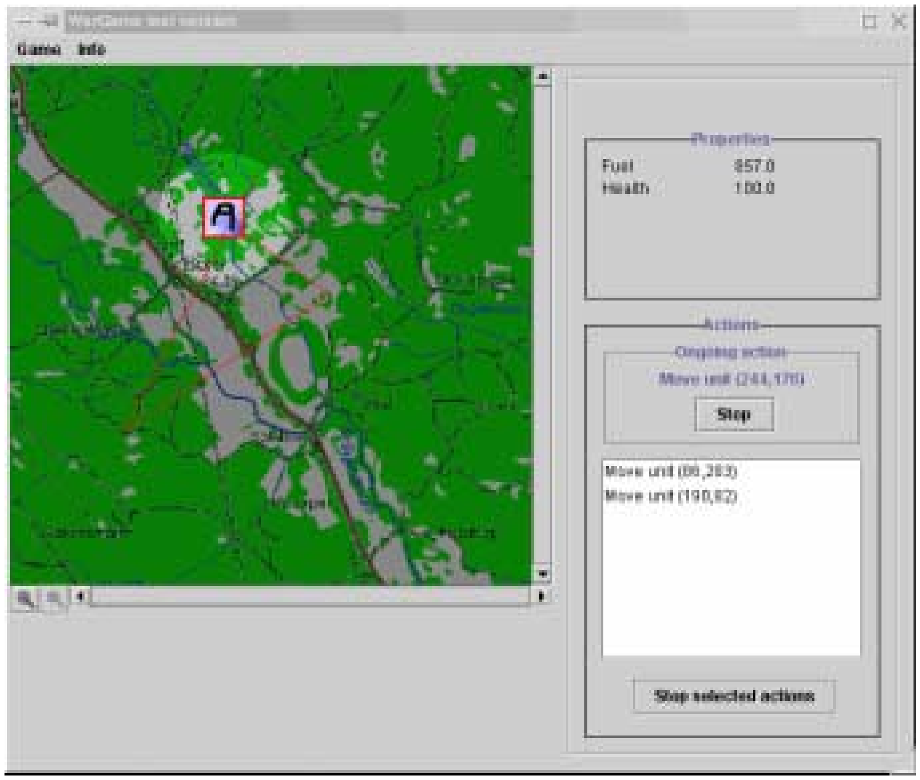
- One wish that will always stay constant is to increase the number of available scenarios.
- Another idea is to create possibilities for communication between clients in order to create a common knowledge between clients in the same "team" regarding for example the ghost units.
- A ghost unit gets less interesting as time passes since the information it represents gets less valuable as time passes. Therefore a time counter should be attached to a ghost unit when the ghost unit is created.

A Resources

The official web page for the GECCO project is available at the following URL.

`http://www.nada.kth.se/theory/gecco/`

B Screenshots



Fuel 857.0
Health 100.0

Move unit (244,170)

Move unit (86,283)
Move unit (190,82)

Figure 1: **2001-04-19 – Early development shot.** This screenshot shows the client moving a unit. The unit is a bitmap and it is currently selected (thereof the red box around it). The ring around the unit shows what the unit can see. The red line is the path on which the unit is moving. You can also see the coordinates the unit moves to on the information panel to the right of the map as well as properties of the unit.

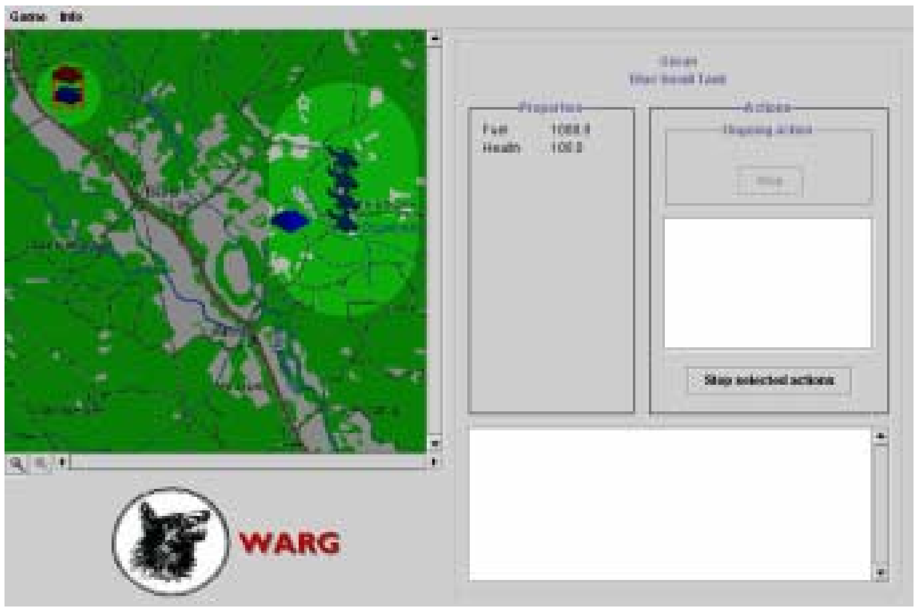


Figure 2: **2001-04-26 – Test game.** The interface has changed slightly from the early development shot. You can also see the new unit graphics with tanks, helicopters and radar stations. For this testgame you can play the red team or the blue team. This picture shows what blue player 1 sees. Because the player controls both the tank and the helicopters it also see the visibility rings for these units. Blue player 1 does not however see what the blue radar station sees. For this game, only blue team leader sees that. The nice graphics for the units are taken from XConq.

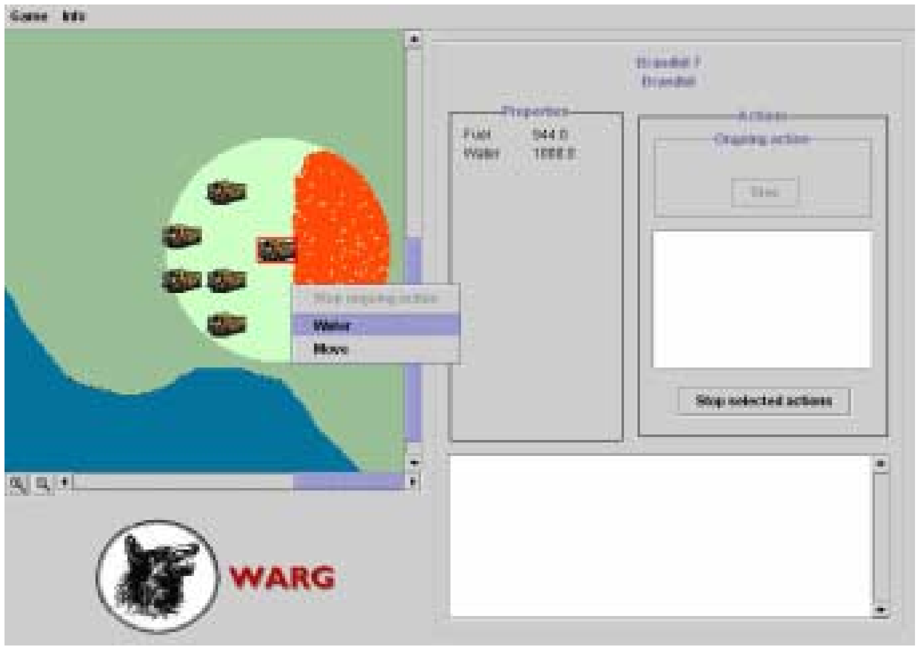


Figure 3: **2001-04-26 – Fire fighter game.** Same engine as test game, but different scenario. This screenshot shows a fire fighter game. The goal is to extinguish the fire (the red in the picture), which isn't so easy as it may seem, because it spreads rapidly across the country and each fire truck only have limited visibility of the fire. In the picture you can also see the action menu, which displays what the unit can do.

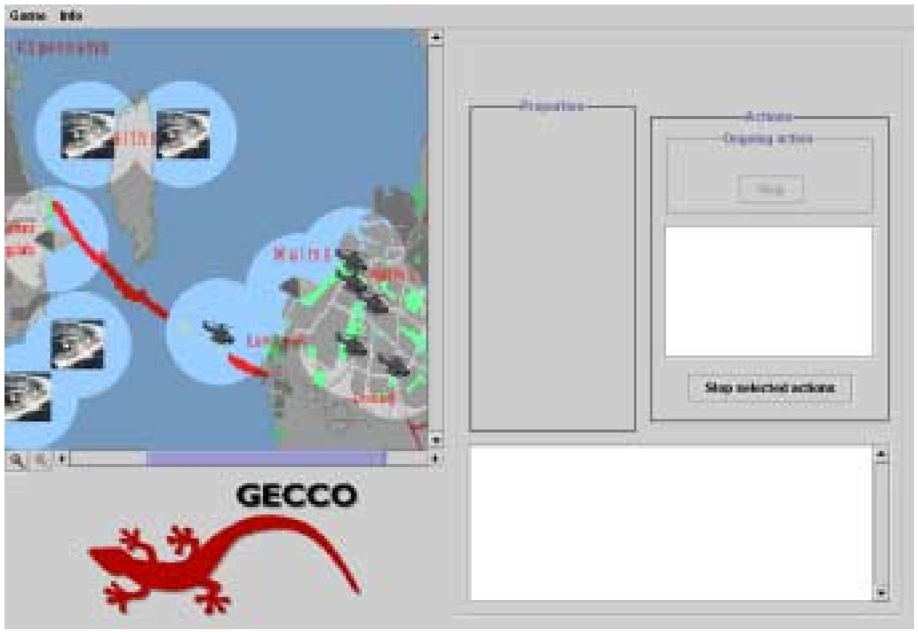


Figure 4: **2001-05-03 – Rescue mission.** A new scenario. This picture shows a game where the Öresund bridge, connecting Denmark and Sweden, has collapsed and you are to rescue the people who are floating around in the water. You have helicopters and boats to your help. The picture shows what the team leader sees, all units and their visibility ranges. The picture also reveals the name change of the game. Our new mascot is a Gecco lizard.

References

[1] David S. Alberts, John J. Garstka, and Frederick P. Stein. *Network Centric Warfare: Developing and Leveraging Information Superiority.* CCRP Publication Series, 2 edition, 1999.

[2] Jan-Ivar Askelin. The catastrophe in Öresund (in Swedish). *FOA-tidningen*, 2:18–19, 1999.

[3] Berndt Brehmer and Dietrich Dörner. Experiments With Computer-Simulated Microworlds: Escaping Both the Narrow Straits of the Laboratory and the Deep Blue Sea of the Field Study. In *Computers in Human Behavior*, volume 9, pages 171–184, 1993.

[4] Berndt Brehmer and Peter Svenmarck. Distributed Decision Making in Dynamic Environments: Time Scales and Architectures of Decision Making. In J. P. Caverni, M. Barhillel, F. H. Barron, and H. Jungermann, editors, *Contributions to Decision Making – I.* Elsevier Science B.V., 1995.

[5] Joel Brynielsson, Henrik Bäärnhielm, Andreas Enblom, Jing Fu Zi, Niklas Hallenfur, Karl Hasselström, Henrik Hägerström, Oskar Linde, Klas Wallenius, and Jon Åslund. *GECCO Developer's Manual.* Department of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden, May 2001.

[6] Joel Brynielsson, Henrik Bäärnhielm, Andreas Enblom, Jing Fu Zi, Niklas Hallenfur, Karl Hasselström, Henrik Hägerström, Oskar Linde, Klas Wallenius, and Jon Åslund. *GECCO User's Manual.* Department of Numerical Analysis and Computer Science, Royal Institute of Technology, Stockholm, Sweden, May 2001.

[7] Joel Brynielsson and Rego Granlund. Assistance in Decision Making: Decision Help and Decision Analysis. In *Proceedings 6th International Command and Control Research and Technology Symposium*, Annapolis, Maryland, July 2001.

[8] Per-Olof Fjällström, Göran Neider, Mats Persson, Tore Risch, and Per Svensson. Architecture Principles for Information Superiority in Future Command and Control Systems (in Swedish). Technical Report FOA-R--00-01435-505--SE, Defence Research Establishment (Sweden), 2000.

[9] John Funge. Cognitive modeling for computer generated forces. In *Proceedings 8th Conference on Computer Generated Forces and Behavioral Representation*, Orlando, Florida, May 1999.

[10] Rego Granlund. *C^3Fire – A Microworld Supporting Emergency Management Training.* Licentiate thesis, Linköping University, 1997.

[11] Rego Granlund. Web-Based Micro-World Simulation For Emergency Management Training. In *Proceedings International Conference on Web-based Modelling and Simulation.* Society for Computer Simulation International, San Fransisco, CA, 1999.

[12] HKV. RMA – A new foundation for defense forces development (in Swedish). Technical Report HKV 09 100:63046, Swedish National Defence, March 1999.

[13] Jan Kuylenstierna, Joacim Rydmark, and Tonie Fåhraeus. A commanders need for information (in Swedish). Technical report, Swedish National Defence College, October 1999.

[14] Tom M. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[15] Mats Persson. Future Command and Control Systems: operational, organisational, and functional prerequisites. In *Proceedings Systemics and Informatics Research Seminar on AIH – Applied Informatics for Improvement of Human Life (SIMS'01)*, Östersund, January 2001.

[16] Claes Sundin and Henrik Friman, editors. *ROLF 2010 – A Mobile Joint Command and Control Concept*. Swedish National Defence College, 1998.

[17] Claes Sundin and Henrik Friman, editors. *ROLF 2010 – The Way Ahead and The First Step*. Swedish National Defence College, 2000.

[18] Klas Wallenius. A Network Centric Info-Structure for the Swedish Armed Forces. In *MILINF 2000*, Enköping, September 2000.

[19] Klas Wallenius. Use of Modern Information Technology: WASP – A Common View of the Situation. In *Technet Europe 2000. 21st AFCEA Europe Symposium and Exposition*, Prague, October 2000.

[20] Rickard Westberg. Decision support for naval command and control systems. Master's thesis, Department of Computer and Systems Sciences, Royal Institute of Technology, Stockholm, Sweden, April 2001.

[21] Per Wikberg. Twelve scenarios for Command and Control (in Swedish). Swedish National Defence College.

[22] Alexander E. R. Woodcock, Derek K. Hitchins, and Cobb Loren. The Strategic Management System (STRATMAS) and the Deployment of Adaptable Battle Staffs. In *Proceedings 4th International Command and Control Research and Technology Symposium*, Stockholm, Sweden, September 1998.