Enhanced Situation Awareness using Random Particles

Joel Brynielsson*
joel@kth.se

Mattias Engblom†‡§
mattiase@kth.se

Robert Franzén†§
d00rface@dtek.chalmers.se

Jonas Nordh†
jonas.nordh@ericsson.com

Lennart Voigt†
lennart.voigt@ericsson.com

*Royal Institute of Technology, SE-100 44 Stockholm, Sweden
†Ericsson Microwave Systems AB, SE-431 84 Mölndal, Sweden
‡Point of contact, phone: +46-70-4241632
§Student

Abstract

Modern command and control systems present the current view of the situation through a situation picture that is being built up from fused sensor data. However, merely presenting a comprehensible description of the situation does not give the commander complete awareness of the development of a situation. This article presents a generic tool for prediction of forthcoming troop movements. The technique is similar to particle filtering, a method used for approximate inference in dynamic Bayesian networks.

The prediction tool has been implemented and installed into an existent electronic warfare system. The tool makes use of the system's geographic information system to extract geographic properties and calculate troop velocities in the terrain which is, in turn, being used for the construction of the tool's transition model. Finally, the result is presented together with the situation picture.

The prediction tool has been evaluated in field tests performed in cooperation with the Swedish Armed Forces in an exercise in Sweden during the spring of 2005. Officers and operators of the electronic warfare system were interviewed and exposed to the tool. Reactions were positive and prediction of future troop movements was considered to be interesting for short-term tactical command and control.

1 Introduction

The commandment of subordinates and the desire to make well-informed decisions has been studied by generals ever since man started to engage in battle. The pioneering work by Sun Tzu is, in several respects, as fundamental today as it has been for the past 2000 years[7]. The nature of the commanders' decision problems has however evolved over time regarding for example the amount of available time and the amount of basic data for decision-making. Today huge amounts of data are being used for rapid decision-making whilst commanders in the past could use days to scratch their heads and think about the data they were missing. Command and control (C2) is a timeless notion that captures both of these epochs:

> **Command and Control** is everything an executive uses in making decisions and seeing that they are carried out; it includes the authority accruing from his or her appointment to a position and involves people, information, procedures, equipment, and the executive's own mind.[3]

In modern C2, where sensors provide vast amounts of information, the *situation picture* is usually thought of as the most important piece of equipment being available in a C2 center. The situation picture consists of a computer-generated map conveying the current view of the situation and, hence, is connected to geographical data and information regarding situation development. The creation of the basic data to be used for creating the situation picture is a non-trivial task denoted data fusion which aggregates available (sensor) information into information that should be conveyed to a decision-maker. Although the situation picture is often thought of as a map merely displaying current unit positions, it is important to notice that higher level fusion within the C2 system also call for prediction of future situation development. This is recognized in level 3, denoted "Impact Assessment", which is the highest information refinement level in the well accepted JDL data fusion model[17]:

> **Impact Assessment** – estimation and prediction of effects on situations of planned or estimated/predicted actions by the participants (e.g., assessing susceptibilities and vulnerabilities to estimated/predicted threat actions, given one's own planned actions).[13]

In this article we describe a generic tool for prediction of forthcoming troop movements, to be used in conjunction with a situation picture in a short-term tactical C2 system. As an example of the tool's usage, it has been implemented and tested in a mobile electronic warfare C2 system. Section 2 discusses the nature of "situation development awareness". Section 3 discusses the computational building blocks and section 4 presents and analyses the algorithm in detail. Section

5 then presents a prototype implementation and the mobile C2 system that has been used as a test bed. In section 6 we present and discuss field tests performed in cooperation with the Swedish Armed Forces. Finally, section 7 concludes and discusses ideas for future work. Details regarding the prototype implementation, referenced in section 5, can be found in appendices A and B in the form of an UML deployment diagram and an XML schema for vehicle specification.

2 Situation development awareness

A common goal of C2 systems is to keep track of and use vast amounts of available information, with varying relevance, in a proper and timely manner to establish situation awareness in order to support planning and decision-making. The establishment of situation awareness is provided through a situation picture that is being built up from fused sensor data, which is being refined into comprehensible information that can be presented to the commanders. However, merely presenting a comprehensible description of the situation does not give a complete understanding of the development of a situation. Also, to not be outmaneuvered a commander must try to estimate future development, i.e., predict what future situation pictures are going to look like. Due to the huge amounts of information, the high tempo, and the complex nature of C2 decision-making[1] it is impossible for a human commander to think of all possible future states. Hence, to obtain full *situation development awareness* the decision support process should assist the commander by predicting opponent plans and suggesting future courses of actions. In military decision-making this state of awareness, referred to as "predictive battlespace awareness", is believed to bring about a shift from a passive discovery mode to a proactive targeting mode[9, 10, 16].

The challenges and difficulties when it comes to prediction are however fundamentally different depending on available time and resources. In this work we focus on short-term tactical decision-making. In such a decision situation, time is of uttermost importance and, due to the short-term tactical decision task, the commanders are in most cases able to make good decisions based solely on the situation picture. Even though the situation picture is indeed important, this does not mean a decision support tool for prediction should not be included in forthcoming C2 systems. On the contrary, considering that most work so far has been directed towards the establishment of the situation picture itself, we believe that a tool for prediction would certainly provide an additional edge.

3 Prediction in tactical command and control

A Bayesian network is a way to represent the full joint probability distribution for a model. The representation is a directed acyclic graph where each node represents a stochastic variable and the topology describes the conditional independence between the variables in the model. The absence of an edge between two nodes means that they are conditionally independent. Each node has a probability distribution conditioned by its parents, so every element in the full joint probability distribution can be calculated as the product of these probabilities. The gain of using a Bayesian network is primarily that the conditional independence can be exploited. This means that no effort will be wasted on probabilities conditioned on variables without effect on the variable's probability.

Dynamic Bayesian networks[8] are used to model domains that change over time. It can be thought of as a number of connected Bayesian networks, each representing a time slice. The variables have dependencies on variables in the current time slice and in the previous ones.

As mentioned above, each element in the full joint probability distribution can be calculated. However, there is significant time to be won if an approximation can be calculated instead. The most suitable method for Bayesian networks is likelihood weighting[12]. The approximation is obtained by first letting any observed evidence variable stay fixed to its observed value. Then the remaining variables are sampled and each sample is weighted by the probability that the sampled event occurs, given the evidence variables. The sum of these weights constitute the approximation.

Particle filtering is a method used for approximate inference in dynamic Bayesian networks and similar structures. It is basically likelihood weighting used on dynamic Bayesian networks. In this case the Markov property of the model can be used, meaning that the state in a time step only depends on a limited number of previous ones. Hence, instead of generating the samples one after the other they can be generated in parallel constituting the state in each time slice. The samples can then be propagated forward to the next time slice using a transition model.

A problem with particle filtering is that the state variables do not depend on the evidence of the new time slice, so the samples are generated without consideration thereof. However, this dependency can be estimated by letting samples be weighted by the probability that the sampled event occurs, given the new evidence, and letting samples with low weight be discarded. This means that only likely samples get propagated to the next time slice, which gives new evidence the desired influence on the approximation.

4 Algorithm description

Our method, which we denote *random particles*, is similar to particle filtering but contains necessary modifications that make the method simpler and slightly different. Basically, the method consists of the particle filtering prediction step, i.e., where the particles are propagated forward using the transition model but without the incorporation of new evidence. In turn, the lack of new evidence makes it possible to treat the particles separately, i.e., instead of propagating the whole probability distribution forward during each time step we iterate over one particle at a time and then start over with a new particle. This gives the system a nice side effect where the algorithm can compute as many particles as time permits resulting in a better target probability distribution. The transition model is given by the unit's properties and the geographical data of the surrounding terrain as described in section 5. The geographical data is obtained from an off-the-shelf geographical information system[2].

Using random particles gives a consistent approximation of the probability distribution. This can be shown by letting $\mathrm{N}(\mathbf{x}_t)$ be the number of particles in state $\mathbf{x}_t$ and N the total number of particles. If we also assume that the algorithm is consistent up to time t we have

$$\frac{\mathrm{N}(\mathbf{x}_t)}{N} = \mathrm{P}(\mathbf{x}_t). \tag{1}$$

The number of particles in state $\mathbf{x}_{t+1}$, i.e., $\mathrm{N}(\mathbf{x}_{t+1})$, is given by the summation of the particles in every state multiplied by the transition model, that is

$$\begin{aligned}
\frac{\mathrm{N}(\mathbf{x}_{t+1})}{N} &= \frac{\sum_{\mathbf{x}_t} \mathrm{P}(\mathbf{x}_{t+1}|\mathbf{x}_t)\mathrm{N}(\mathbf{x}_t)}{N} \\
&= \frac{N \sum_{\mathbf{x}_t} \mathrm{P}(\mathbf{x}_{t+1}|\mathbf{x}_t)\, \mathrm{P}(\mathbf{x}_t)}{N} \\
&= \sum_{\mathbf{x}_t} \mathrm{P}(\mathbf{x}_{t+1}|\mathbf{x}_t)\, \mathrm{P}(\mathbf{x}_t) \\
&= \mathrm{P}(\mathbf{x}_{t+1}),
\end{aligned}$$

where the first step is given by equation (1) and the last step is given by the definition of conditioning. Hence, the approximation is consistent as long as we have a correct prior distribution $\mathbf{P}(\mathbf{X}_0)$.

Random particles can be used to calculate the probability distribution of a unit's future location. In this application each particle represents a simulated unit. The properties of the particle are its geographical position and a speed vector. The pseudo code for the algorithm is presented in algorithm 1.

When the particles are initially created, there exists an observed position and possibly a speed vector depending on what kind of sensor provided the information. A radar sensor, for example, can detect a movement because of its accuracy and high refresh rate. A radial movement can be detected thanks to the phase shift. A direction finding unit, on the other hand, can only supply a position due to its lack of accuracy and inability to detect units while they are not transmitting.

The transition model $\mathbf{P}(\mathbf{X}_{t+1}|\mathbf{x}_t)$ consists of two parts, one part for the direction of the unit's movement and one part for the speed of the movement. For both parts, a matrix provided by the terrain speed operator in the geographical information system SpatialAce[2] plays a decisive role. This matrix contains the maximum speed the specified type of vehicle can keep in the surrounding terrain.

The model for the direction is built up from a Gaussian distribution with mean 0. The mean is given by the assumption that a unit is most likely to continue to travel in its current direction. The standard deviation affects how willing the unit is to change its current course. What is left to do on the direction transition model is to update it with respect to the surrounding terrain. This is done by multiplying the probability for each turn angle by the speed the unit can keep in that direction, times a factor representing the influence of the surrounding terrain. The speed is given by the terrain speed operator mentioned above. After normalizing the distribution a sample can be taken in order to turn the unit.

The second part of the transition model is a probability distribution given by the specification of the unit's type. From this distribution a sample can be taken. The sample is then compared to the maximum possible speed given by the terrain speed operator, and the unit's speed is set to the smaller of the two.

Now we got a complete transition model and the particle can be propagated forward.

The algorithm consists of two nested loops. The outer one is executed once for each new particle until the desired precision is achieved or for as long as available time permits. The inner loop is executed once for each time step.

Inside the inner loop the probability distribution is updated and normalized and two samples are taken, one for the course change and one for the speed. If the length of the probability distributions are assumed to be fixed this gives a time complexity of $\mathcal{O}(T)$ for the inner loop. This coupled with the N executions of the outermost loop gives a total time complexity of $\mathcal{O}(NT)$.

When it comes to space complexity one has to take into account that the algorithm keeps the current particle and all the particles simulated before that in memory. It also stores a matrix with all the maximum speeds according to the terrain. The size of this matrix depends on how fast the unit can travel, how far into the future one wants to predict and the grid size. In the worst case, the matrix will have a size of $(2hv_{max}/s) \times (2hv_{max}/s)$ where v_{max} is the maximum speed, h is the time horizon of the prediction and s is the grid size.

Algorithm 1 Position prediction using random particles

input:	*originalParticle*, the particle representing the observed unit h, how far into the future to predict Δt, the time step N, the number of particles $\mathbf{P}_{turn}$, the turn probability distribution $\mathbf{P}_{speed}$, the speed probability distribution $\mathbf{V}$, a matrix containing the maximum speeds allowed according to the geographical environment f, a factor determining how much the geographical environment affects $\mathbf{P}_{turn}$
output:	$\mathbf{P}_{position}$, the probability distribution for the unit's position with the same size and resolution as $\mathbf{V}$
local variables:	$T \leftarrow \lceil h/\Delta t \rceil$, the number of time steps to predict $\mathbf{currentParticles}$, a vector holding N particles, initially N copies of *originalParticle*

do until desired precision or as long as time permits
 for $j \leftarrow 1$ to T
  if the unit's direction has not been observed
   $\mathbf{currentParticles}[j].direction \leftarrow$ random direction
  Reset $\mathbf{P}_{turn}$ to the original distribution
  for each turn direction φ
   Multiply $\mathbf{P}_{turn}[\varphi]$ by $f \times \mathbf{V}[position]$, where $position$ is the future position for a unit travelling in direction φ
  NORMALIZE($\mathbf{P}_{turn}$)
  turn $\mathbf{currentParticles}[j]$ by a sample from $\mathbf{P}_{turn}$
  sample *newSpeed* from $\mathbf{P}_{speed}$
  $\mathbf{currentParticles}[j].speed \leftarrow$ MIN($newSpeed, \mathbf{V}[position]$) where $position$ is the future position for a unit travelling in the current direction
  move $\mathbf{currentParticles}[j]$ a time step according to the new speed and direction
register the position of each particle in *currentParticles* by adding 1 to the position in $\mathbf{P}_{position}$ that corresponds to the square the particle is located in
NORMALIZE($\mathbf{P}_{position}$)

If we once again assume that the size of the probability distributions held in memory is fixed we get a space complexity of $\mathcal{O}(N + (2hv_{max}/s)^2)$. Since h is proportional to T and if we assume that v_{max} and s are fixed (v_{max} is given by the specification of the unit and s by the resolution of the geographical information) we get a space complexity expressed in only N and T, namely $\mathcal{O}(N + T^2)$.

5 Target command and control system

The target C2 system is part of the electronic warfare (EW) system Galder, a product of Ericsson Microwave Systems AB. The Kosovo Force (KFOR) is using the predecessor of this system for peacekeeping purposes in former Yugoslavia. The system is made up of a number of direction finding and electronic countermeasure units which can be seen in figure 1. Figure 2 illustrates the operator's workplace in the predecessor of Galder. The tasks of an EW system are electronic support (detection, direction finding, interception) and electronic attack (spoofing, jamming). The information obtained by the EW system is forwarded to a mobile C2 system along with information from other types of sensors such as radars.

The C2 system is built as client-server software, where the client's main purpose is to build and present the situation picture while the server communicates with the other direction finding and electronic countermeasure units and collects data from a variety of sources, such as the direction finding sensor. The design is illustrated in an UML deployment diagram in appendix A.

The client has been built using Microsoft .NET technology and is split up into three main modules; presentation, logic and communication. Furthermore, it contains two support modules, namely a data and a control module. The objective of the presentation module is to present the graphical user interface (GUI) as seen in figure 3. The communication module is a connection point between the client and the server, whereas the logic module works as a logical unit in between the communication and the presentation module. Common data structures and constants are enclosed in the data module and common GUI controls in the control module.

The server consists of a database for persistent storage, a database functions module for handling the database (DBF), a message handler for communication within a unit, unit to unit communication and a set of functional interfaces, such as the interface used for communicating with the direction finding sensor. The message handler software is reused from an AEW&C system (Airborne Early Warning & Control System) named Erieye[5].

Galder uses an advanced toolkit, SpatialAce, for building interactive geographic applications. SpatialAce consists of four main parts; a core set of components which handle reading, presentation and manipulation of geographical data, different programming interfaces (APIs) that make it possible to interact using a

Figure 1: The direction finding and electronic countermeasure units in Galder.

variety of programming languages and two tools for creating and viewing map configuration files[2].

Galder interacts with SpatialAce using COM components via COM-interops for the .NET framework. The interaction is handled completely inside the presentation module. The map is a layered system where each layer can contain various types of data, for example roads, cities, forest and application specific data. Each layer may be visually turned on and off at the will of the operator. A layer is a part of a well-defined data flow model. At the end of the data flow we find a dataset, which is an abstraction of the physical data location, whether it be a shape file on disk or an in-memory store populated by the application. At the beginning of the data flow we find a view object, which represents the section of the world that the application is currently interested in. Connected to the view are several layers. Between each layer object and the datasets any number of operators can be connected. The operators are similar to filters and in some way manipulate the geographical data that flow through. For example, an operator could visualize the data or read specific data from a dataset[2].

Figure 2: The operator's workplace.

Our solution, the prediction component, is implemented as a software component, according to the software component definition of Clemens Szyperski[15], in the .NET language C#. The component is integrated in the C2 system via the presentation module. Also, the component is dependent on SpatialAce and the map configuration file used by Galder.

The procedure of performing a prediction starts when an operator selects an object on the map of interest. Thereafter, the object type is selected, for example a tank, and a time horizon and a precision parameter of the prediction is specified. The precision parameter affects the number of particles and the length of each time step. If the current speed and direction of the object is known, these parameters can also be specified. The prediction can now be initialized via the interface, `IPrediction`, provided by the prediction component. First, the component retrieves a matrix of the area of interest from a SpatialAce terrain speed operator, using the arguments specified. The matrix describes the maximum speed at which the specified object can travel within squares on the map. The resolution, i.e., the size of each square, of the calculations is limited by the resolution of the provided map. Our component has been tested on maps with a pixel resolution of 10 by 10 meters and 50 by 50 meters. The difference can be seen in figure 4. Second, the matrix is passed to the algorithm, which is independent of SpatialAce, and

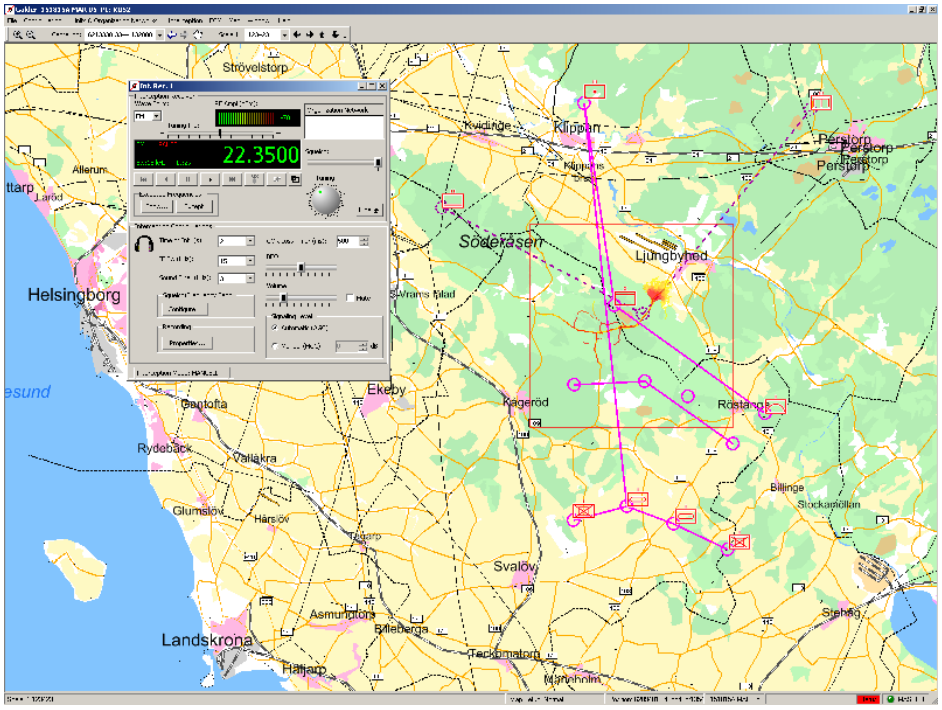


Figure 3: The graphical user interface of Galder.

after calculations a prediction matrix is returned. Finally, this prediction matrix is thereafter presented on a layer on the map of the target C2 system.

The component uses two layers in order to perform its work. One layer is used to calculate the maximum speed of a vehicle on specific areas of the map, while the other is used to present the resulting prediction on the situation map display. The speed layer contains two operator chains, one for calculating the speed of an object on different terrains using the terrain speed operator provided by SpatialAce and one for calculating the speed of the object on different types of roads. The maximum values of the output from these two operator chains are then merged to the resulting speed matrix. The reason for using two operator chains is due to that roads are not taken into account in the terrain speed operator.

In order to understand how the maximum speed matrix is derived we will now describe the specification of an object and how that is used in the two operator chains. Our work also included creating a component for reading and specifying objects. The objects or vehicles are specified in an XML file according to an XML schema, which can be seen in appendix B. Each object has a specified name, water speed, maximum head-on slope, maximum side slope and maximum snow depth. In addition to these parameters it also contains a terrain speed, a road speed and

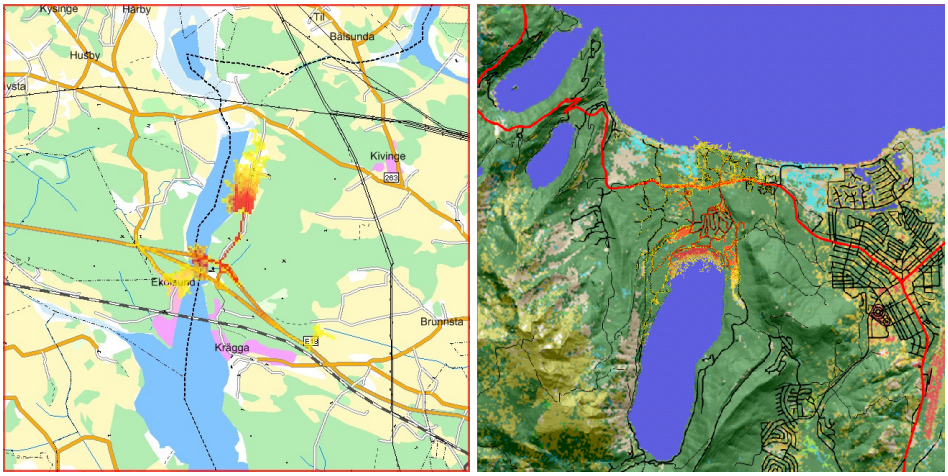


Figure 4: The difference between map grid sizes of 50×50 and 10×10 meters per pixel.

a speed distribution specification. The terrain speed specification is used by the terrain speed operator. In short, the specification contains a set of four-tuples. Each tuple describes at what maximum speed an object can travel given a ground condition, a surface structure and a slope parameter[11]. The map can, using this set of four-tuples and a specified vehicle, be transformed from describing the terrain to describing a speed matrix.

The specification of a vehicle also includes the road speeds, which describes at what speed the vehicle can travel on a specified type of road. The operator chain calculating the speed matrix uses this information to transform the road information into a speed matrix. As described earlier, the road matrix and the terrain speed matrix is processed through a maximization operator and the result in each square is the maximum of the two matrices.

Finally, the vehicle specification contains a speed distribution. The purpose of this specification is to add the possibility of specifying possibilities of an object traveling at a specified speed. For instance, a military vehicle has a higher probability of traveling at a slower speed when traveling within a military column than when isolated. Furthermore, the speed distribution adds the possibility of modeling an object that not necessarily constantly travels at its maximum speed.

6 Field tests

The year is 2005 and the country Götaland has been separated into two fractions, the east side and the west side, which in several ways has set the country in an

unstable state. The government of Götaland has requested aid from the European Union (EU) and the United Nations (UN) in order to sustain law and order and to reinstate government control. The Swedish Armed Forces will contribute to the EU forces that will be sent to Götaland. Other countries contributing to the EU forces are Finland, Germany, United Kingdom, Denmark and The Netherlands. The main objectives of the mission are to disarm the irregular units of the fractions, reopen the border between the east side and the west side and see to that a supervised and controlled election of parliament is performed[14].

This was the scenario being used in a military exercise with approximately 6000 participating men and women engaged by the Swedish Armed Forces during spring 2005. The purpose of the exercise was to give the soldiers basic training in planning, supervising and executing tasks within peace-keeping and peace-promoting missions. The training included a wide spectrum of tasks and events in conflict confrontation. The exercise aimed to be the starting point for discussions regarding how the Swedish Armed Forces is going to educate its officers and conscripts in the future to meet its goals. The goals are to have a force that is able to rapidly accomplish tactical ground control inside a mission area in order to contribute to stability and safety[14].

The exercise included both rural and urban battles. During these battles the collected information presented by the C2 system was massive and to keep track of the troop movements was close to impossible. Under those circumstances we had the opportunity to present our work to people from the Swedish Armed Forces, the Swedish Defence Research Agency and the Swedish Defence Materiel Administration. From this presentation we received feedback and ideas for future work.

Feedback given by the operators of the C2 system pointed out that their task today is merely to report what has happened, not to make predictions about the future. However, they do think that the functionality of our system is well suited for commanders at a higher level.

One of the interesting uses of the prediction function, regardless of command level, is to determine whether a detected unit is the same as a unit that has been detected earlier. This is particularly interesting when the units are transmitting using hop frequencies since they are assigned a new unique fingerprint on each position they occur and, thus, are hard to distinguish. On the contrary, considering units using fixed frequencies for unencrypted transmission this need is not as obvious as it is possible to get hold of call signs by bugging, which can be used to distinguish units from each other. By combining observations with predicted units to determine if they originate from the same unit the algorithm comes even closer to particle filtering. The new observations become the new evidence that causes unlikely samples to be filtered out. This also gives a possibility to disregard old obsolete observations from the situation picture in Galder.

Another issue pointed out during the field tests is that the prediction functionality also can be used on friendly units, e.g., it can be used to find rendezvous points for a number of units. It can also help finding possible routes through the terrain that the operator might have overlooked.

7 Conclusions and future work

The demonstrations performed during the field tests attracted much attention. Reactions were, in general, positive and prediction of future troop movements was considered to be very interesting for short-term tactical C2. The feedback came mainly from personnel engaged by the Swedish Armed Forces, working at different levels in the C2 hierarchy, ranging from the operators of the EW system to skilled commanders with experience from operative C2 at division level. Experienced commanders at higher level C2 seemed to show more interest in using some kind of prediction tool compared to the operators of the EW system itself. We believe this is due to the nature of their respective jobs where an operative level commander is more focused on what will happen in the future while the EW system operators are more concerned about what has happened so far. However, with the inclusion of multiple observations, as mentioned in section 6, both operators and experienced commanders meant the solution would be of interest to operators as well since it helps them to interpret the past.

Another extension mentioned during the field tests is to include a warning system. Such a system should be able to warn the commander if a hostile unit could constitute a threat to a specific point or area within a certain time period.

Except for the extensions mentioned above, we believe future work should be concentrated mainly towards visualizing the result. As of today we obtain a sound probability distribution regarding the likelihood of a vehicle being at a certain position, but we are still uncertain regarding how one best conveys this result to a commander. In the prototype system all traversed map positions were highlighted on a scale ranging from yellow, denoting few traverses, to red, denoting intensive use. The problem with this representation is that it does not really represent the resulting probability distribution after a certain amount of time although resulting in a surprisingly interesting picture which, according to military personnel at the exercise, is probably useful. During the field tests, however, the present system gave us the idea to implement a system that, for each time step, stores all particles. With such an implementation, one could easily develop a tool giving the commander the opportunity to visualize the development of the situation 1) continuously updated as time goes by, 2) after T minutes, 3) at $T = 0$ (i.e., outdated but accurate), and 4) using predictions ranging from T_0 to T giving the same result as in the present system.

Recently, our *random particles* have also caught interest from developers of a C2 simulator which is currently under development at Ericsson Microwave Systems AB. The idea is to use the transition model to make simulated units behave in a natural manner.

Acknowledgements

Implementation and field tests have been sponsored by Ericsson Microwave Systems AB within the scope of two Master's theses being written by two of the authors[4, 6]. All photos are taken by Lars Moell, Swedish Defence Research Agency, except for figure 2 which is taken by Ivar Blixt, Försvarets Bildbyrå.

We wish to gratefully acknowledge all people that gave us help and feedback during the field tests. In particular, we wish to mention Lieutenant Colonel Mikael Lundin and Captain Jonny Krantz, Swedish Armed Forces, Hans Elmegren and Bo Rörstrand, Swedish Defence Materiel Administration, and Torbjörn Frostemark, Swedish Defence Research Agency, for their valuable feedback. Furthermore, we would like to thank Ronnie Johansson, Royal Institute of Technology, for commenting on this work.

References

[1] Berndt Brehmer. Dynamic decision making: Human control of complex systems. *Acta Psychologica*, 81(3):211–241, December 1992.

[2] Carmenta AB. *SpatialAce 4.3 Documentation*, Gothenburg, Sweden, 2004.

[3] Thomas P. Coakley. *Command and Control for War and Peace*. National Defense University Press, Washington, District of Columbia, 1991.

[4] Mattias Engblom. Position prediction using random particles. Master's thesis, Royal Institute of Technology, Stockholm, Sweden, 2005.

[5] Ericsson Microwave Systems AB. Erieye Airborne Early Warning & Control (AEW&C) system product datasheet. `http://www.ericsson.com/microwave/pdf/erieye.pdf` (accessed March 12, 2005).

[6] Robert Franzén. Component-based software development in mobile command and control systems. Master's thesis, Chalmers University of Technology, Gothenburg, Sweden, 2005.

[7] Ooi Kee Beng and Bengt Pettersson, editors. *Sun Tzu: The Art of War (Swedish translation of the original text)*. Swedish National Defence College, Stockholm, Sweden, second edition, 1999.

[8] Kevin Patrick Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley, 2002.

[9] Paul W. Phister, Jr., Timothy Busch, and Igor G. Plonisch. Joint synthetic battlespace: Cornerstone for predictive battlespace awareness. In *Proceedings of the Eighth International Command and Control Research and Technology Symposium*, Washington, District of Columbia, June 2003.

[10] Robert A. Piccerillo and David Brumbaugh. Predictive battlespace awareness: Linking intelligence, surveillance and reconnaissance operations to effects based operations. In *Proceedings of the 2004 Command and Control Research and Technology Symposium*, San Diego, California, June 2004.

[11] Mikael Rittri. *Cross-Country Analysis with SpatialAce*. Carmenta AB, Gothenburg, Sweden, October 2004.

[12] Stuart J. Russell and Peter Norvig. *Artifical Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, New Jersey, second edition, 2003.

[13] Alan N. Steinberg and Christopher L. Bowman. Revisions to the JDL Data Fusion Model. In David L. Hall and James Llinas, editors, *Handbook of Data Fusion*, chapter 2. CRC Press, 2001.

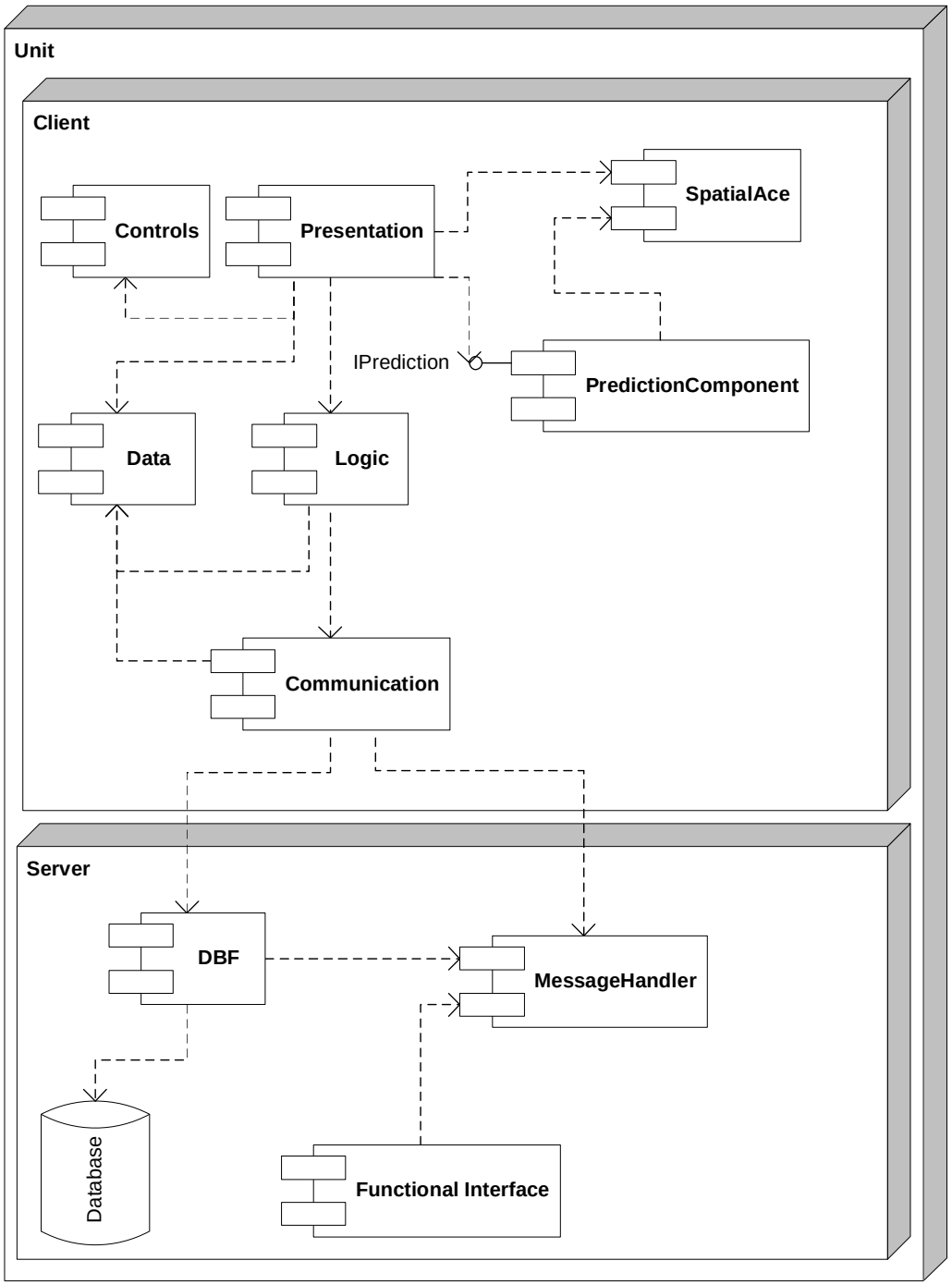
[14] Swedish Armed Forces. Operation Götaland: Description of the final Swedish army exercise 2005 (ASÖ05). `http://www.armen.mil.se/article.php?id=13269` (accessed March 12, 2005).

[15] Clemens Szyperski. *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley/ACM Press, New York, second edition, 2002.

[16] US Air Force. *Aerospace Intelligence Preparation of the Battlespace*. Air Force Pamphlet 14–118, Headquarters, Department of the Air Force, Washington, District of Columbia, June 2001.

[17] Franklin E. White. Managing data fusion systems in joint and coalition warfare. In Mark Bedworth and Jane O'Brien, editors, *Proceedings of EuroFusion98 International Conference on Data Fusion*, pages 49–52, Great Malvern, United Kingdom, October 1998.

A UML deployment diagram of the C2 system



B XML schema for vehicle specification

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema targetNamespace="http://www.ericsson.com/microwave/vehicletypesschema.xsd"
  elementFormDefault="qualified"
  xmlns="http://www.ericsson.com/microwave/vehicletypesschema.xsd"
  xmlns:mstns="http://www.ericsson.com/microwave/vehicletypesschema.xsd"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:complexType name="LandSpeed">
    <xs:sequence></xs:sequence>
    <xs:attribute name="GroundCondition" type="xs:short" />
    <xs:attribute name="SurfaceStructure" type="xs:short" />
    <xs:attribute name="Slope" type="xs:short" />
    <xs:attribute name="Speed" type="xs:double" />
  </xs:complexType>
  <xs:element name="vehicleTypeList">
  <xs:complexType>
    <xs:sequence maxOccurs="unbounded">
      <xs:element name="vehicleType">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="landSpeeds">
              <xs:complexType>
                <xs:sequence maxOccurs="unbounded">
                  <xs:element name="landSpeed" type="LandSpeed"></xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="roadSpeeds">
              <xs:complexType>
                <xs:sequence maxOccurs="unbounded">
                  <xs:element name="roadSpeed">
                    <xs:complexType>
                      <xs:sequence></xs:sequence>
                      <xs:attribute name="roadCode" type="xs:int" />
                      <xs:attribute name="speed" type="xs:int" />
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
              <xs:attribute name="key" type="xs:string" />
            </xs:complexType>
          </xs:element>
          <xs:element name="speedDistribution">
             <xs:complexType>
                <xs:sequence maxOccurs="unbounded">
                   <xs:element name="speedProbabilityPair">
                      <xs:complexType>
                        <xs:sequence></xs:sequence>
                        <xs:attribute name="speed" type="xs:double" />
                        <xs:attribute name="probability" type="xs:double" />
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="name" type="xs:string" />
            <xs:attribute name="waterSpeed" type="xs:double" />
            <xs:attribute name="slopeMax" type="xs:double" />
            <xs:attribute name="sideSlopeMax" type="xs:double" />
            <xs:attribute name="snowDepthMax" type="xs:double" />
```

```
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```