# Adaptive Support of Human Attention Allocation using Cognitive Models

Teun Lucassen

Master's Thesis

**Abstract:**

This research investigates the support of human attention allocation. A fixed support system is compared with two forms of support which are adaptive to the user by using cognitive models. A liberal and conservative variant of the adaptive support are introduced. The goal of the support is to improve the task performance of the user during a tactical picture compilation task. Although the results of the conducted experiment have not shown a significant improvement in task performance when adaptive support is given, the negative effects of inappropriate reliance seen in fixed support where no longer present in the adaptive condition.

This page was intentionally left blank.

# Preface

This research was done as a graduation study for the Master Human Media Interaction at the University of Twente in cooperation with TNO Human Factors in Soesterberg. An internship within the same project was done prior to this study.

I would like to thank Peter-Paul van Maanen for his supervision over this study on behalf of TNO. Kees van Dongen has been a very supportive team member with valuable input. I would also like to thank Dirk Heylen for his supervision on behalf of the University of Twente as the first supervisor, along with the other members of the examination committee, Betsy van Dijk and Anton Nijholt.

My three roommates at TNO, Roald Dijkstra, Maarten Hoeppermans and Iwan de Kok have provided the necessary distraction during the period in which the research was done, along with all other interns. This has resulted in a "secondary research" on the prediction of the appearance of hot snacks at the cafeteria. The results were promising, but the external validity seemed insufficient for a research publication.

Finally, I would like to thank my fiancée Martine Schiphouwer for providing a listening ear when I came home after work, especially at the times when the research was not satisfactory.

This page was intentionally left blank.

# Table of contents

# 1. Introduction

## 1.1. Background

One of the trends seen in the naval warfare domain is a decreased manning. This means that the same tasks have to be performed by less people. Also, the complexity of several tasks is increasing, due to both an increase of the available information and an increase in complexity of the environment [Grootjen et al. 2006]. These observations result in an increased work load for military personnel.

When being stressed with a high work load operators tend to make more errors in their tasks. Attention has to be divided amongst several tasks and several items within a task, leaving only a small amount of attention for each task or item.

Errors may appear with both novice and experienced users [Pavel et al. 2003], since the attentional resources of a person will always be limited, despite exhaustive training in the task at hand [Wickens 1984, Kahneman 1973]. The consequences of errors are often quite severe in warfare.

This research focuses on the reduction of problems caused by errors in attention allocation. Three types of support models are introduced to assist the user in spreading his attention over all items which are important for the task execution in an optimal fashion.

In this research the support is focused on a tactical picture compilation task (TPCT) in the naval domain. A digital radar is presented on which operators have to assess the threat levels of the various contacts (ships) on the screen based on given criteria. The five most threatening contacts have to be selected. Since the contacts move over the screen, the selection has to be updated regularly to achieve a good performance.

The proposed support systems could also be applied in various other domains and tasks, such as air traffic control or ground warfare. The task at hand should contain a fairly large number of objects amongst which the attention of the operator should be divided.

Well allocated attention is important for a good task performance. In the case of a naval ship the task performance might be the decisive factor between life and death.

## 1.2. Research goals

In preceding studies on this subject [Koning et al. 2008, Lucassen 2008] cognitive models of attention were developed and validated which are able to:

1. describe where the focus of attention of the user is (descriptive);
2. prescribe where the focus of attention of the user should be (prescriptive).

These models can be used in the development of adaptive support systems. The output of these models could directly be presented to the user in some fashion. Other cues from the environment and the user might also be incorporated to contribute to the performance of the support systems. Examples of these cues are mouse clicks or other actions by the user.

An adaptation model is designed and implemented to support users in the allocation of their attention. Attention levels for the objects in a task are found by determining discrepancies between the descriptive (as it is) and prescriptive (as it should be) model of attention. The output of the model is a sequence of actions in a test environment that is aimed at changing the user's allocation of attention from the current (descriptive) to a desired state (prescriptive). This can for instance be done by making the objects which require attention visually different. Early studies have not succeeded in doing this.

An initial adaptive system has already been developed [Koning et al. 2008]. The conducted pilot experiments did however not yet show an improvement in task performance when the support is used. The developed adaptive support systems in this research should yield an increased task performance.

## *1.3. Research questions*

The motivation to use adaptivity in the support models is that it is likely that the support becomes less interruptive and more pleasant to work with, since it will not disturb the operator when it is not necessary. When the system knows that the operator is doing his task right, no shifts in attention will be necessary.

The main comparison made in this study is between the task performance of a user with and without adaptive attention allocation support. Two variants of adaptive support are introduced: a conservative and a liberal system. The difference between these two systems is the influence of the user on the support. The conservative system takes the user as much as possible into account, where the liberal system relies more on itself. This should result in a system which only gives advice when really needed. For example, if a system is only adaptive on the attention allocation of the operator, it might try to divert the attention to an object which does not have any attention. We call this liberal adaptive, since it is adaptive to attention allocation. However, there might be other evidence that this object does not need attention, such as mouse clicks of the operator near or on this object. The addition of this extra evidence to the support model is referred to as conservative adaptivity.

In order to assess the influence of adaptivity in the support, a third support type is added as a baseline. This is a support system which is not adaptive to the user. The task performance of the user with this fixed support should be higher than without any support. Otherwise, there would be no motivation to introduce any support. On the other hand, problems inflicted by the lack of adaptivity should be made clear. The expected problems are inappropriate over and under reliance on the support.

The research questions can be summarized as follows:

1. Can attention allocation support help to improve the task performance (both fixed and adaptive)?
2. Which problems are inflicted by fixed support?
3. How can adaptivity contribute to an increased task performance compared to no support and fixed support?
4. What is the influence of a liberal/conservative setting on the task performance?

## *1.4. Hypotheses*

The conditions are abbreviated for easy reference:

1. NS – No Support
2. FS – Fixed, non-adaptive Support
3. LAS – Liberal Adaptive Support
4. CAS – Conservative Adaptive Support

When looking at the LAS and CAS conditions together, it is abbreviated to AS.

Based on the research questions in Section 1.3, the hypotheses are stated as follows.

### 1.4.1. Task performance with fixed support

The task performance of the user should be better when using fixed support than without any support. This is needed to show that the addition of some type of attention support is useful to improve task performance.

Hypothesis 1: The task performance of the user with FS is better than with NS.

The fixed support condition is needed to show that the influence of adaptive support on the task performance is caused by adaptivity instead of the support system as a whole.

### 1.4.2. Inappropriate reliance on fixed support

An expected problem when fixed support is offered is inappropriate reliance. Users may rely too much or too little on the available support, possibly resulting in a lower task performance than optimal.

When the accuracy of the support is low, but the user has over-reliance on the support, there is a higher probability that the advice is followed, also when it is incorrect. This results in a lower task performance.

When the accuracy of the support is high, but the user has under-reliance on the support, advice is likely not to be followed, even when it is correct. This will also result in a lower task performance.

This effect is especially expected when the accuracy of the support varies over time. We expect to make two observations:

1. When the performance of the support decreases from a good performance, the task performance will drop below their own task performance when performing the task without support. This is caused by over reliance based on experience with a well-functioning system.
2. When the performance of the support increases from a poor performance, the expected increase in task performance will delay for some time. The user will need a moment to figure out that the performance has increased and will inappropriately under rely on the system.

The variation in levels of accuracy in the support is proposed in Figure 1.

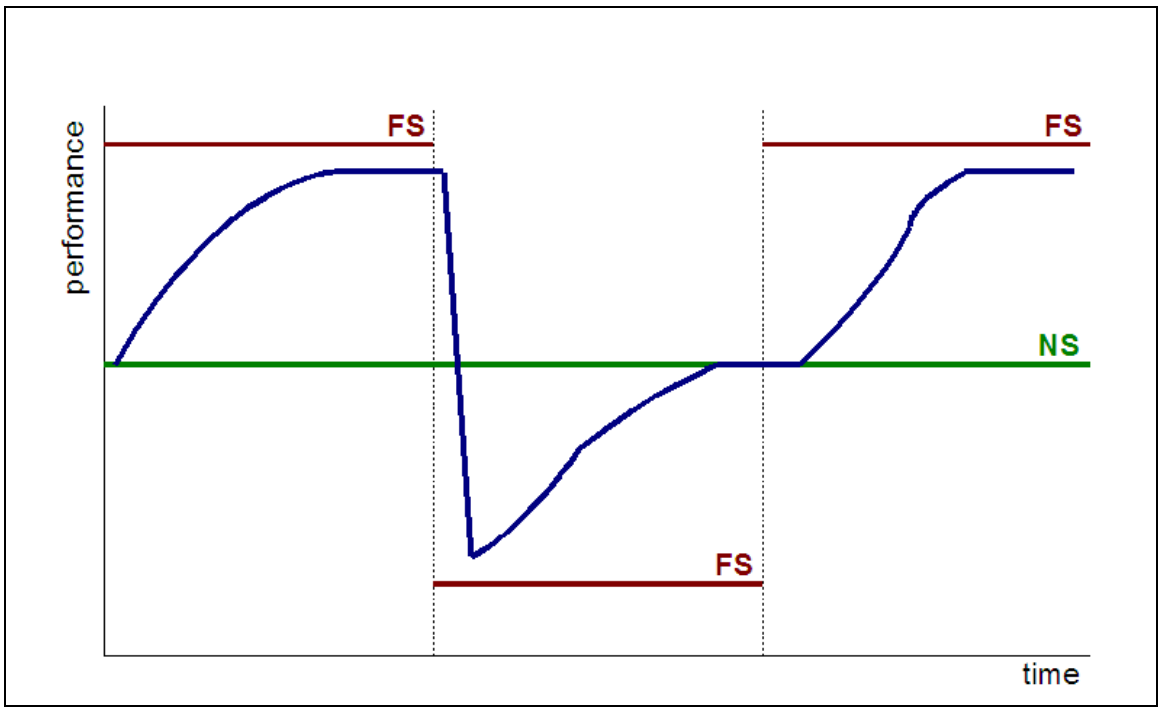



**Figure 1: Variations in support accuracy**

The red lines show the high and low performance support levels. The blue line shows the expected task performance of the user. The green line is the intrinsic task performance which a user can achieve without support (NS).

Hypothesis 2: Fixed support causes inappropriate reliance.

This can be demonstrated when all of the following hypotheses below are true. These three sub hypotheses cover the different signs for inappropriate reliance.

Hypothesis 2a: Advice is followed, also when the support accuracy is low.

This can be observed by measuring the task performance during the various support accuracy levels in the fixed support condition. When these task performances significantly differ from the task performance without support, the support has an impact on the participant.

Hypothesis 2b: Users are sensitive for changes in the support accuracy.

This can be shown by measuring the task performances during various levels of support accuracy. When these levels significantly differ, the changes in support accuracy have an impact on the participant.

Hypothesis 2c: Users adapt their behavior to the changes in the support accuracy.

A delay in the adaption will occur when the support accuracy changes. This delay is caused by memory effects, learning effects and complacency effects. During the delay, the reliance on the support is inappropriate. When this delay occurs, inappropriate reliance is demonstrated. The delay can be demonstrated by looking at the relative task performance during the first and second half of an interval with a certain support accuracy, after a change in support accuracy.

### 1.4.3. Reduction of inappropriate reliance by adaptive support

When the hypothesis in section 1.4.2 is true, it acts as a motivation to introduce adaptive support. The same analyses can be performed for both adaptive conditions. The negative effects of inappropriate reliance on task performance should then be reduced.

Hypothesis 3: The usage of AS reduces the inappropriate reliance in the case of FS.

The adaptive support keeps the user more in the loop (kept up-to-date, see section 2.3.3). This means that the user relies more on himself and has more situation awareness (being aware of what is happening around you). When the support accuracy drops, the user is already in the loop and capable of performing the task without proper support.

Figure 2 shows the expected effect of the NS, FS, and AS conditions on the task performance.

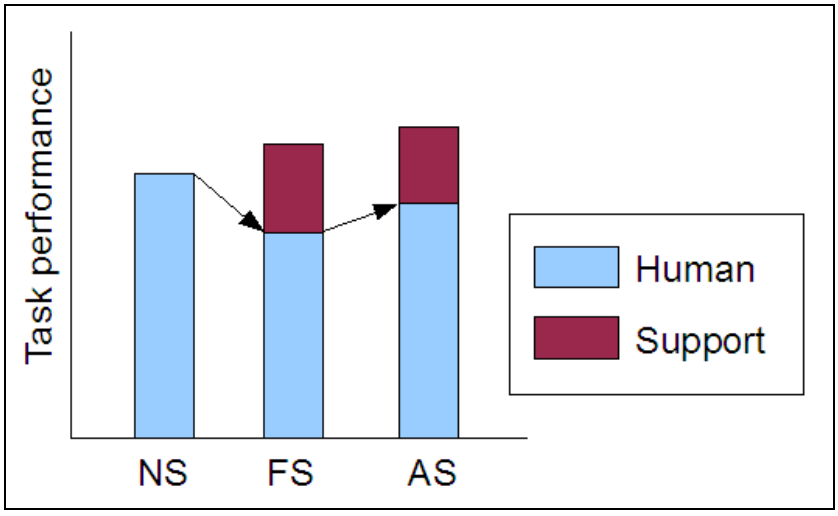



**Figure 2: Effect of being support types on task performance**

The task performance is in both support conditions partly caused by the performance of the human and the performance of the support (the blue and red bar). In the fixed support condition, the task performance of the human is hampered by the fact that he is (partly) out of the loop by the support. This drop in performance is compromised by the influence of the support. In the adaptive support condition, the drop of the human part of the task performance should be less. It might be that the influence of the support in the adaptive condition is also less than in the fixed conditions, but this is compensated by the improved human task performance.

### 1.4.4. Task performance for good and poor performers

For well performing users, the task performance should increase with the use of adaptive support, since the user has more influence on the support. For poor performers, the fixed support should yield better task performance.

Hypothesis 4: Users with a high task performance benefit more from the adaptive support then users with a low task performance.

Users which are able to perform the task very well without support, due to personal talent or affectivity with the task have a high intrinsic task performance. When a user has a high intrinsic task performance and he has more influence on the support, it is expected that the resulting task performance is higher.

### 1.4.5. Conservative and liberal setting

The conservative setting is more adaptive to the user compared to the liberal setting. This results in a system which will only try to divert the attention of the user when strictly necessary according to its task model. This keeps the work load demanded by the support system as low as possible, expecting to result in a higher task performance.

Hypothesis 5: The task performance of the user with CAS is better than with LAS.

## *1.5. In this document*

A literature study is performed to found the principles of the adaptive system. Hereafter, the theory behind the support model and its implementation is described. The method for validation is treated, along with the results. At the end of this document conclusions are drawn from the obtained results.

# 2. Literature review

In this chapter, related work to this study is discussed.

## *2.1. Human attention allocation*

Several aspects of human attention allocation are important when trying to improve it using adaptive support.

### 2.1.1. Overt/covert

An important distinction between types of attention is its status. Attention can be overt or covert [Gibson 1974]. Overt attention is the process where the focus of attention is directed towards a certain stimulus. When the attention is covert, the person is mentally focused on the stimulus, assessing its properties. In the Tactical Picture Compilation Task (TPCT), overt attention is needed to allocate attention to the contact. After this, covert attention is needed in order to assess the threat level of the contacts.

The support system is only able to support overt attention, since its goal is to direct the focus of attention to the contacts for which attention is required according to the support system.

### 2.1.2. Bottom-up/top-down

When attention is drawn to a certain stimulus, this can be caused by bottom-up or top-down processes [Conner et al. 2004]. A process is bottom-up when the stimulus itself stands out in the environment in such a manner that attention is automatically drawn to it. An example is a bright red square amongst several dark blue circles. The saliency of the stimulus is the decisive factor. The more salient a stimulus is, the bigger is the chance that attention is bottom-up drawn to this stimulus.

Attention can also be directed by top-down processes. In this case, a person voluntarily directs his attention to a stimulus. This can for example occur, when a person is instructed to search for certain properties of a stimulus, such as a square amongst circles, triangles and other shapes. When the other properties of the stimuli (such as size, color, and luminance) also vary, the person has to assess all stimuli on the desired property. The intended stimulus in the TPCT task does not pop-out visually which means that the attention has to be directed top-down.

### 2.1.3. Problems in attention allocation

One of the problems seen in attention allocation is change or inattentional blindness [Mack and Rock 1998]. It occurs when a significant change in the current situation occurs without being noticed by the attendee.

In this research the user can for instance be focused on some less important stimuli (contacts) while some other contacts become significantly more threatening. The fact that such a change is not noticed by the user might have disastrous consequences in the naval warfare domain.

Another problem is the over allocation and under allocation of attention to certain stimuli. When all available stimuli need to be monitored, the limited attentional resources need to

be divided amongst them. Different stimuli might require a different amount of attention due to its properties (e.g. its variability over time). When a stimulus receives more attention than it requires due to its properties, the attention is over allocated. When a stimulus receives less attention than required the attention is under allocated.

Over allocation of attention for a certain contact might occur when the user suspects that it will become more threatening in the near future. When the user stays focused at this contact, but its threat level does not rise, the attention for this contact is over allocated. Due to the limited attentional resources, over allocation of attention for one contact implies under allocated attention for other contacts.

In the TPCT task, well allocated attention is very important. A lot of obects need to be assessed in order to make the correct selection. It is expected that any loss in performance in attention allocation is directly visible in the task performance.

## 2.2. Task

Several tasks have been used as cases to show the effects of adaptive automation. The radar task is very similar to the task of an air traffic controller (ATC), but also in the field army domain, similar tasks (such as monitoring the environment, based on GPS or other information) exist. This does not only increase the amount of research already done on this subject, but also increases the value of the results of this research.

The task that is supported in this research is monitoring a digital radar and assess for each of the contacts on the radar whether they are threatening or friendly. It is also known as the tactical picture compilation task (TPCT). Another example of a TPCT in the naval domain is [Heuvelink 2006], which focuses on reasoning on the acquired data. This task remains interesting because it is yet virtually impossible to be executed by computers. The operator needs to interpret the actions of possible enemies and predict what they will be doing in the future. However, it is possible to assist humans in the execution of this task to increase their performance. This is the main focus of this research.

The tactical picture compilation task shares a lot of characteristics with the multiple object tracking task (MOT). It is known that humans can track 4 or 5 individual moving objects [Pylyshyn 2001, Pylyshyn and Annan 2006]. This means that in the TPCT task, the attention of the user has to shift between objects from time to time. It is in these shifts that errors are likely to occur. The user has to make a selection somehow of what area (or objects) to focus their attention on at what time. An exhaustive overview of the systems of this control of attention is given in [Wickens 2007].

## 2.3. Support systems

### 2.3.1. Automation and attention allocation

Support of attention allocation can be seen as a form of automation. The support system does not take over the overall task from the user, but some subtasks are taken over by the system. In this research one of the subtasks is the allocation of attention. A proper allocation is needed in order to perform the overall task well.

Four classes of subtasks (or functions) can be distinguished [Wickens and Hollands 2000, Inagaki 2003]:

1. Information Acquisition
2. Information Analyses
3. Decision selection
4. Action Implementation

The adaptive support type is automation in the information acquisition class. Some data is filtered out and the attention of the user is drawn to contacts which require this. The assessment of these contacts is however completely left to the user.

The fixed support acts as a support during the decision selection. The support given to the user equals the task that the user has to perform. This means that the user has the opportunity to follow the support in all cases whilst not assessing the objects himself. Errors in the support will also be followed.

It is because of this inappropriate reliance that unreliability in the support is likely to be more costly (in terms of task performance) for the fixed support than for the adaptive support [Rovira et al. 2002(I), Rovira et al. 2002(II)]. This difference is however strongly task dependent [Galster and Parasuraman 2004].

### 2.3.2. Saliency

Several modalities are possible as a communication channel of the support to the user. The task is strictly visual, but other modalities may be considered to offer support to the user. Several studies [Sarter et al. 2000, Sklar and Sarter 1999] have shown the advantages of multi-modal interaction. Especially when the usage of visual cues is not salient enough, other modalities such as auditory or tactile feedback might be used. Auditory cues (e.g. spatial) are effective to decrease search times for visual cues [Bolia et al. 1999].

The support system itself should however not consume too many resources from the user. When the support becomes too salient, it would be hard for the user to focus on the task itself and the work load will rise. The user should be able to finish his current assessment before attending to the support. Otherwise, the user would be interrupted in his task execution, which yields worse performance [Bailey and Konstan 2006].

The note that users should be able to “ignore” the support for some period means that the shifts made in focus of attention remain voluntary. This means that the user can decide for himself whether to follow the advice or not. When the support is too salient, it is very hard to ignore which results in involuntary shifts. Given the fact that the performance of the support will never be perfect, it might lead to a decrease in task performance, since the user cannot ignore incorrect advice. It is suggested that attention capture by visual cues is always voluntary [Remington et al. 2001]. This means that the user is always able to react to a visual stimulus at the time he wants to.

Independent of the form the support is given in, it is important that the manipulation of the stimuli matches the top-down settings of the user [Theeuwes and Chen 2005].Top-down settings can be described as the form of manipulation that is expected by the user. This can either be achieved by a very logical, salient cue to draw attention to a certain

contact or by supplying clear instructions to the user about what he can expect from the support.

### 2.3.3. Support problems

One of the issues that can be addressed by attention allocation support is change (inattentional) blindness. The support can divert attention to changes in the environment which are not noticed by the user.

The problem of over and under allocation of attention is also an issue which can be taken into account in support systems. With knowledge about the current focus of attention of the user, the support can detect that attention levels for certain contacts are inappropriate and divert attention to other contacts.

As mentioned earlier, the performance of the attention allocation support will not be perfect. This has multiple reasons.

In a simulated environment, the knowledge of the support about threat levels of contacts can be perfect. However, in a realistic scenario, this will not be the case. Some criteria for the assessment of threat levels can not directly be measured by a computer. An example is the influence of cultural aspects (such as local holidays) or operator experience (such as certain movements from hostile ships). How accurate a computer is able to measure threat levels is unknown, but we assume that the support performance will be in the same range as human performance. On one hand, the computer is able to more accurately measure certain criteria (such as speed or distance). On the other hand, some criteria might not be incorporated in the prediction of support systems.

When the support system makes mistakes, this will highly influence the trust and acceptance of the user [Parasuraman and Riley 1997, Dzindolet et al. 2003]. The reliance of the user on the system will be affected. This reliance is likely to be inappropriate when the performance of the system varies over time. Suppose a support system has posed a well performance for some time. When suddenly the performance drops, the user is likely to over rely on the support. The opposite also applies. When the support performance has been poor for some time, the user is likely to under rely on the support when its performance rises.

Another problem that might occur while supporting attention allocation is the difference between novice and expert users. Especially novice users will profit from the support, since they have not worked out a personal approach for the task execution. Expert users may be hampered by the support, since the manner in which the task is approached by the support system might differ from their personal approach [Beilock et al. 2002].

An issue that needs to be considered is the out of the loop effect of the user. When the user has the opportunity to just follow the support system instead of making his own decisions on threat levels, the user might get out of the loop. This is not desirable for two reasons. First, when the situation occurs that support is no longer given (for example caused by technical difficulties), it might take some time before the user is back in the loop and able to perform the task accurately by himself. Second, the situation awareness is critically hampered when the user is less in the loop [Endsley and Kiris 1995]. Situation awareness is very important for a high task performance, especially in the naval domain.

Adaptive automation is also influential on the situation awareness of the user. When decisions are made by the automated system, this might decrease the situation awareness [McClernon et al. 2006]. An example of a computational model of situation awareness is [McCarley et al. 2002]. Situation awareness is vital in the naval warfare domain.

Some critics reckon that automation based on the skills of machines and humans (MABA-MABA) does not work since the division of work is quantitative and the effects are qualitative [Dekker and Woods 2002].

### 2.3.4. Existing support systems

An example of a support system which takes the attention of the user into account is the Saab Driver Attention Warning System [Saab 2002]. Field tests are performed in which the support system constantly monitors the driver of a car. The system will alert him when any signs of drowsiness of fatigue are detected. The advantage of monitoring the driver instead of his actions (e.g. abrupt direction changes) is that the system is able to react earlier, preventing accidents.

The results of studies on the human element in marine accidents [Itoh et al. 2004, Psaraftis et al. 1998] serve as a motivation to introduce support systems in this domain. Most examples of existing support systems focus on collision and grounding avoidance. The situation awareness is being raised by offering more information to the operator, such as the location of surrounding ships and GPS information. The consequences on the cognitive load of operators are investigated in [Lee and Sanquist 2000].

Cognitive models have been used in support systems in domains very similar to the naval domain, such as aviation [Taylor 2001, Taylor et al. 2002, Wickens et al. 2001], air defense [Santoro and Kieras 2005] and control [Fisher et al. 1999], and ground battlefield [Horrey and Wickens 2001]. In [Roda and Thomas 2005] an exhaustive overview of attention aware systems is given.

# 3. Support models

This chapter describes the fixed and adaptive support models which are later implemented and tested in an experiment.

The support models are applied in a task where the user has to make a selection of a number of objects in a larger pool of objects, based on their priority. When support is given to an object, it means that the support system tries to reallocate the focus of attention of the user to this object.

## *3.1. Fixed support model*

The design of the fixed support system is quite straight forward. It should yield the best task performance of the user without using any information about the user (such as gaze using an eye-tracker or other user actions).

During an earlier experiment one of the questions in the last questionnaire was in which way they would want to be supported, given that a support system is available. Most participants wanted the support system to do the task for them. The participants could check the solution of the system on its correctness. This is an evident solution.

The fixed support system can make a suggestion which objects have a high priority. The user can accept or decline this solution, or alter it.

Note that when the user only partially follows the advice, he can and has to assess the incorrect parts of the solution himself.

## *3.2. Adaptive support model*

### 3.2.1. Input cues from the environment

The developed cognitive model [Koning et al. 2008] is adaptive to the user in because it takes the gaze of the user into account in the decision whether to give support or not. This type of adaptivity should contribute to the performance of the support system, since it only gives support to the user when certain objects or areas are not attended. One can imagine that when a user has already assessed an object in the task, it would be highly inconvenient when the support system tries to draw attention to this object again.

It is possible to directly translate the cognitive model to a certain type of support in the test environment, for example by varying the luminance of the objects. Illuminated objects draw the attention of the user bottom-up, since they are visually significantly different from non-illuminated contacts.

Other information about the user could also be incorporated into the model for improvement. Next to gaze information extracted from an eye-tracker, the actions of the user in the test environment are already available without the need for extra sensors. Some examples of these actions are mouse movements, clicks, and the current state of task execution by the user. The system could also assess the hits, misses, false alarms, and correct rejections that a user makes, keeping in mind that the system is not able to estimate the correct solution perfectly in a realistic scenario. Another option is to keep

track of the mental workload of the user [Harris et al. 1993, Hilburn et al. 1997, Di Nocera et al. 2006] and adapting the support to this workload.

### 3.2.2. Design of the adaptive support model

Various cues from the environment can be used to contribute to the performance of the attention allocation support system, such as information about mouse movements or other user actions. Two options are discussed here: support based on false alarms and misses and support based on the solution of the user.

*Option 1: Support based on false alarms and misses*

This option assumes that support is only needed when a user makes a mistake in the task. Users can make two types of mistakes:

1. False alarms (FA)
2. Misses (MISS)

The support system will only support the user when a contact has no attention from the user and is a false alarm or a miss according to the system.

Figure 3 shows an example of a user solution at some point (with 2 FA's and 2 MISS). Only those objects which are marked as false alarms or misses are assessed on their attention level. When this level is low, the contacts will be offered as support.

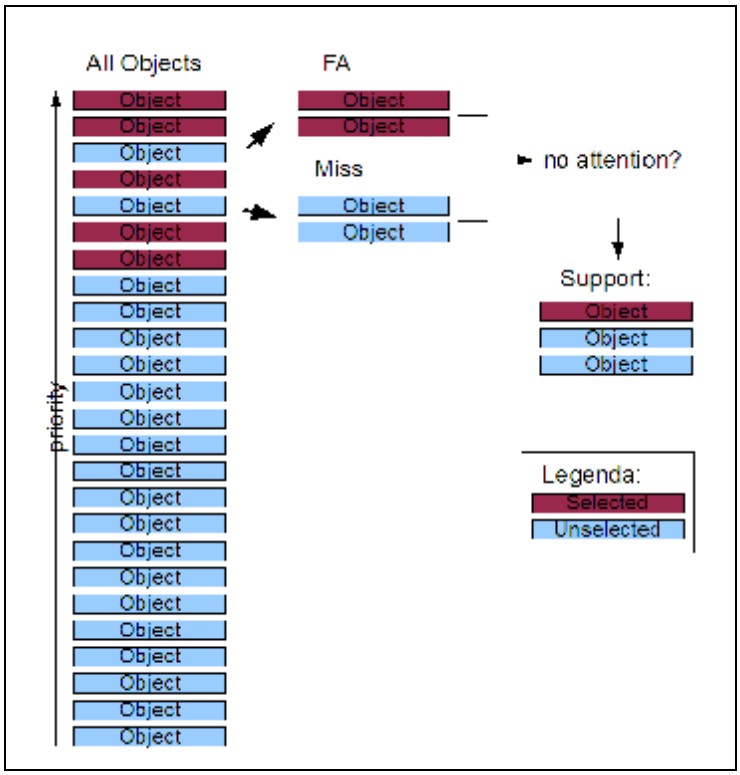



**Figure 3: Support based on false alarms and misses**

Note that the user has made two mistakes in the top five most threatening contacts. This results in two false alarms and two misses. In this example, only one of these four contacts has the attention of the user, so support is given to the other three.

In order to create a conservative and liberal setting an α-value could be introduced which represents the size of the fraction of the objects in the final selection which are actually supported. In a liberal setting, α could be 1. In a more conservative setting, α could for example be 0.5. In a setting without support, α is 0. The objects in the final selection would have to be ranked on their priority to illuminate the most important objects in settings where $\alpha < 1$.

*Option 2: Support based on selection*

An interesting cue to keep into account is the selection each user makes. Objects which are part of the current selection require a different attention allocation strategy than unselected objects.

A selected object has to be assessed on the possibility of deselection due to a decreased priority. Unselected objects have to be assessed when they have a rising priority.

This means that attention is required for selected objects with a relatively low priority and unselected contacts with a relatively high priority.

Figure 4 shows an example of the selection of the objects to support.

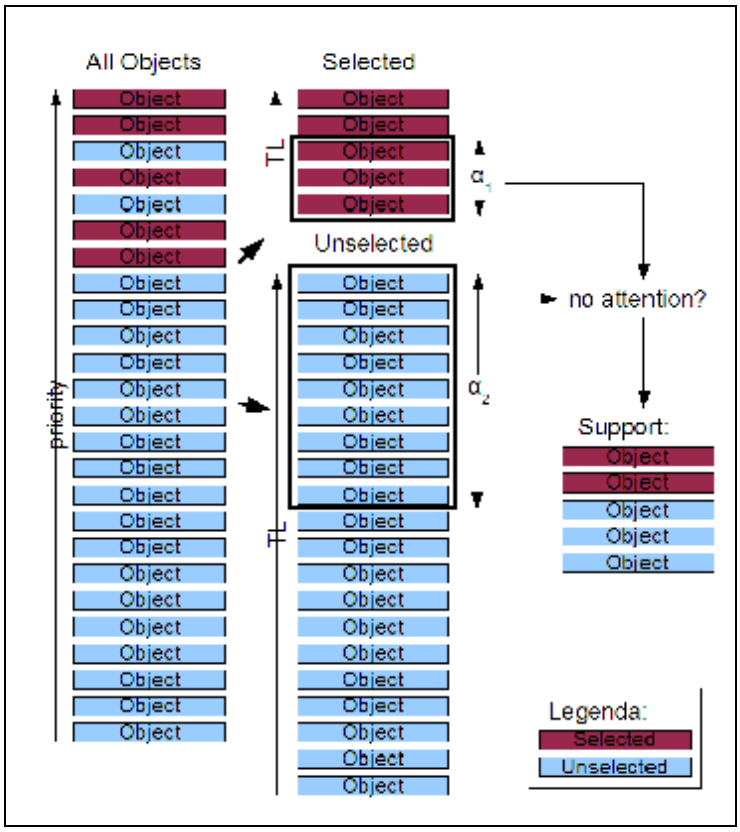



**Figure 4: Support based on selection**

Note that the first two columns are ranked on priority.

The selection of the user in this case is the same as in Figure 3. The number of selected and unselected objects that are picked from the second column are dependent on $\alpha_1$ and $\alpha_2$. These values can be determined using the task performance data of the first

experiment. From this data, we can derive the average number of correctly selected (and unselected) objects, along with the standard deviation.

The liberal setting is implemented by using only the first two columns of Figure 4. This means that all objects in $\alpha_1$ and $\alpha_2$ are supported. The conservative setting only supports the objects which do not have attention according to the cognitive model.

*Discussion*

Both options are adaptive to the users' actions. The main (and essentially only) action a user has to perform in this task is to select and deselect objects. Both systems are adaptive to this selection. Option 2 uses information about the selection directly. Option 1 uses information about false alarms and misses, which are a direct result of the selection.

The system in option 1 is essentially the same as in option 2, but now with a variable $\alpha$, adaptive to the number of false alarms and misses. This means that option 1 will only help when a mistake in the task has already been made (a false alarm or a miss). Option 2 will help to stay focused on those objects which have to be attended by the user since they are nominated for (de)selection. Unselected contacts which have a very low priority will never be supported. Neither will selected objects with the highest priority.

The second option is preferred. Option 1 only supports the user when an error has already been made. Option 2 helps the user to allocate his attention to the objects which require this to prevent errors.

# 4. Implementation

In this section the implementation of the support models, along with the necessary manipulation are described. Appendix A contains the full source code for the most important part of the adaptive support.

## *4.1. Task*

### 4.1.1. Description

The task that participants of the experiment have to perform can be described as a tactical picture compilation task (TPCT). A simulated digital radar screen has to be monitored. The "own ship" is located around the center of the screen. It does not move during the task. It is represented by a blue circle. The area between the green lines is marked as "sea lane". The light gray areas are land. Figure 5 shows a screenshot of the task environment.

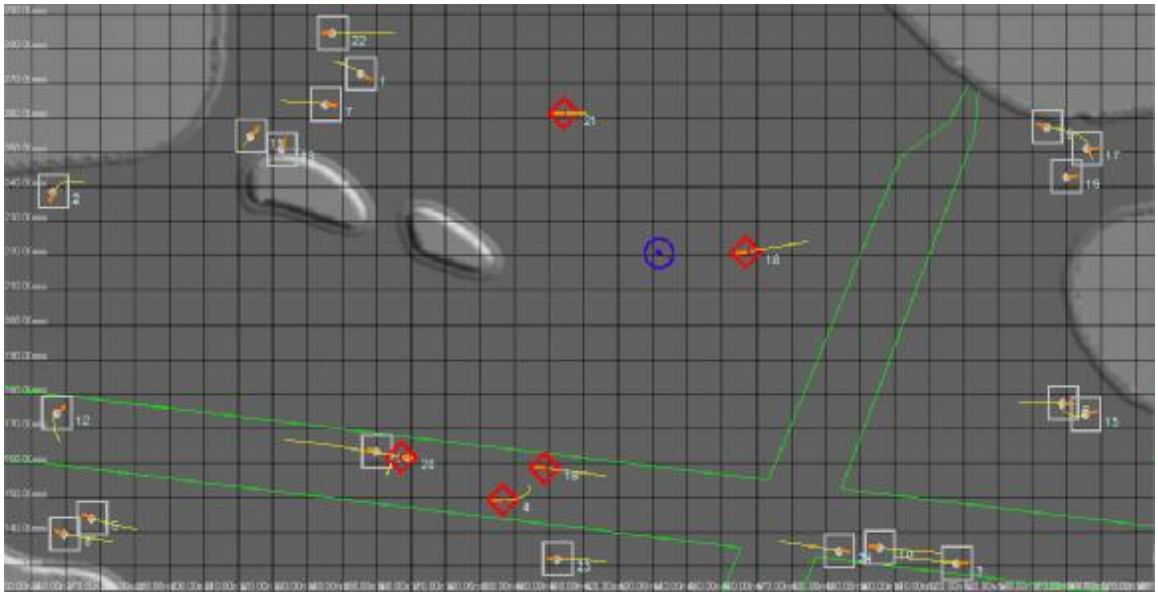


**Figure 5: Screenshot of the task environment**

There are 24 other ships (contacts) present on the digital radar screen. These are initially represented by white squares with a number between 1 and 24. The bold orange line in front of each contact corresponds with its heading. The thin yellow line behind each contact shows the history of the contact. The length of this line represents the speed at which the contact travels; a short line indicates a low speed and a long line indicates a high speed.

The task is to constantly have the five most threatening contacts selected. The threat levels of the contacts are based on four criteria:

- Speed (higher is more threatening)
- Distance to own ship (closer is more threatening)
- Heading (towards own ship is more threatening)
- In/out of the sea lane (out of the sea lane is more threatening)

All criteria are equally important. The number of criteria on which a contact is threatening determines the threat level. When comparing two contacts with an equal threat level, the contact which poses its criteria more clearly is most threatening. For example: one contact is only threatening on speed and another on distance. When the difference in speed between these two contacts is greater than the difference in distance, speed is dominant and thus the first contact is most threatening.

Contacts can be selected by clicking on them. The white square then changes to a red diamond. When the same contact is clicked again, it changes back to a white square.

Because of the movements of the contacts, their threat levels change over time. This means that the selection of five most threatening contacts has to be updated. During an update, the user can either first select an additional contact and then deselect an already selected contact or first deselect a selected contact and then add another contact to the selection. Either way, the user will have to make one mistake because for a short period of time, four or six contacts are selected instead of the required five. The user is instructed that he is free to choose from both options, but recommended to keep the period in which too much or too little contacts are selected minimal.

### 4.1.2. Implementation

The test environment was implemented using the game development tool Game Maker. The ship seem to move over the screen in a random fashion, but they actually follow a pre-defined path.

All contacts can be in two modes: on a turn or not on a turn. Contacts which are not on a turn follow a relatively unthreatening path, which might seem as random to the participants. When a contact is on a turn, it will take on a more threatening path, as if it were to attack the “own ship” in the center, or pose some other threat, such as leaving a sea lane. One to five contacts can be on a turn, which lasts one to three minutes.

A turn is also called a scenario section. Two different scenarios were developed in [Lucassen 2008], a simple and a complex one. The actual perceived difference in difficulty turned out to be minimal. Both scenarios consist of ten scenario sections, but scenario sections can be removed to shorten the experiment.

The scenarios were developed by manipulating the ambiguity and the dynamics of the scenario of the tactical picture compilation task. Concerning ambiguity, small differences in the threat level of contacts were made so that it is more difficult to identify the five most threatening contacts. Dynamics was manipulated by varying the number of threat level changes of contacts over time. Changes in the threat level were such that the number of times that the contacts need to be re-evaluated was relatively high in the complex scenario.

For more details on the implementation of the test environment and particularly the scenarios, see [Lucassen 2008].

## 4.2. Visualization

Several methods can be adapted to visualize the output of the various support models to the user. It is important that all support types use the same type of visualization, so only the support models are compared and not the visualization type.

The most obvious modality to represent the support is the usage of visual cues. Other modalities could also be used, such as audio or tactile feedback. An important factor in the decision on the implementation is the desired dominance of the support. The support should be salient enough to draw bottom-up attention. When the support is too dominant, it might cost too much work load of the user. If the user is constantly interrupted without being able to at least partially ignore it, the task performance might decrease drastically. The risk of the support being too dominant is significant multi-modal feedback. The usage of other modalities than visual might not be desirable for the same reason.

Within the visual domain, several options are available. Some examples are:

1. Varying colors
2. Varying shapes
3. Varying luminance
4. Blinking/not blinking

Varying colors and/or shapes might result in a very confusing interface, where a lot of instructions are needed to let the user appropriately do his job. Even with well instructions, the interface might cost too much work load for optimal task performance. The same goes for making contacts blink; this might be too salient and too interruptive, since blinking is a form of abrupt onset [Jonides and Yantis 1988]. The user should be able to complete his current assessment before diverting his attention to the next for optimal performance.

Regarding the preceding conclusions, the luminance change is an appropriate way to divert attention. Illuminated contacts draw the attention, were other contacts are faded to a lighter tint. An early pilot has shown that the visualization should be discrete instead of continue. When all contacts are assigned some continue value for the visualization, the differences between them are in some cases not enough. This might result in a task shift: instead of assessing threat levels, users now have to distinguish the various support levels. This task might be just as hard as the original task. This observation leads to the decision that the support should be discrete: a contact is either supported or not.


a. b. c. d.

**Figure 6: Unselected and selected contacts**

Figure 6 shows an unselected non-illuminated contact (a.) and an unselected illuminated contact (b.). On the right are a selected non-illuminated (c.) and a selected illuminated contact (d.).

## 4.3. Fixed Support

With the task described in section 4.1 and the visualization in section 4.2, the implementation of the fixed support is the illumination of the five most threatening contacts, leaving the rest to be more transparent.

The algorithm to select the contacts to illuminate is given in Figure 7. It is performed at each timestamp, constantly re-assessing the illuminated contacts.

```
foreach Contact c
        if c.isInTop5MostThreateningContacts
                c.illuminate;
        end
end
```

**Figure 7: Fixed support algorithm**

This implementation implies that if the advice is entirely followed, the digital radar screen shows five illuminated red diamonds and 19 non-illuminated grey squares. This is shown in Figure 8.

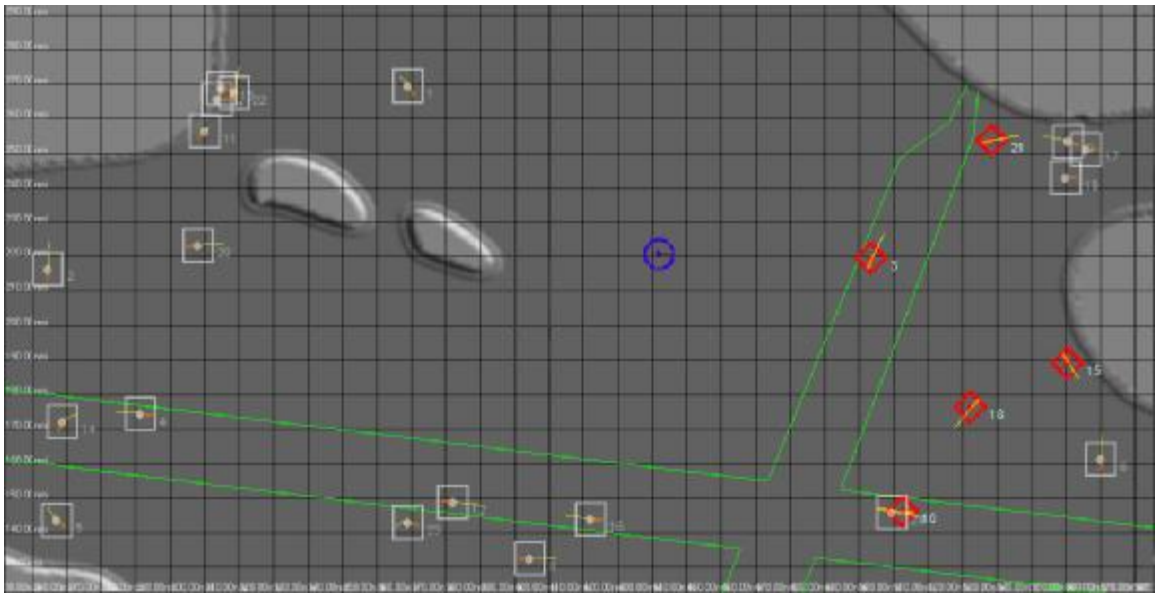


**Figure 8: Screenshot of the task environment with fixed support**

The main problem with this type of support is that the system has incomplete knowledge about the threat levels of the contacts (see Section 4.6). This means that the system is not entirely sure that the five suggested contacts are in fact the five most threatening ones. Some errors will be made by the system in the suggestions.

## 4.4. Adaptive support

### 4.4.1. Liberal adaptive support

The liberal adaptive support described in section 3.2 is implemented by illuminated the two least threatening selected contacts (since they are eligible for deselection) and the three most threatening unselected contacts (since they are eligible for selection). The algorithm in Figure 9 is used to select the contacts to be illuminated.

```
illuminatedSelectedContacts = 2;
illuminatedUnselectedContacts = 3;

foreach Contact c
        if c.isSelected
                if (c == min(selectedContacts)  && illuminatedSelectedContacts > 0)
                        c.illuminate;
                        selectedContacts.remove(c);
                        illuminatedSelectedContacts--;
                end
        else // c is not selected
                if (c == max(unselectedContacts) && illuminatedUnselectedContacts > 0)
                        c.illuminate;
                        unselectedContacts.remove(c);
                        illuminatedUnselectedContacts--;
                end
        end
end
```

**Figure 9: Liberal adaptive support algorithm**

Note that the contacts which are selected to be illuminated are removed from the contact lists after illumination. This is done to select the contact with the second highest or lowest threat level.

The numbers of illuminated selected and unselected contacts are chosen such that the total number of illuminated contacts is equal to the fixed support condition. Since there will be 19 unselected contacts and 5 selected contacts, the number of illuminated selected contacts is lower than illuminated unselected contacts.

### 4.4.2. Conservative adaptive support

In the conservative adaptive support, the contacts selected by the algorithm in Figure 9 are only illuminated when they have no attention of the user according to the cognitive model of attention. This results in the algorithm shown in Figure 10.

```
maxIlluminatedSelectedContacts = 2;
maxIlluminatedUnselectedContacts = 3;

foreach Contact c
        if c.isSelected
                if (c == min(selectedContacts)
                    && maxIlluminatedSelectedContacts > 0)
                        if !c.hasAttention
                                c.illuminate;
                        end
                        selectedContacts.remove(c);
                        maxIlluminatedSelectedContacts--;
                end
        else
                if (c == max(unselectedContacts)
                    && maxIlluminatedUnselectedContacts > 0)
                        if !c.hasAttention
                                c.illuminate;
                        end
                        unselectedContacts.remove(c);
                        maxIlluminatedUnselectedContacts--;
        end
end
```

**Figure 10: Conservative adaptive support algorithm**

Note the difference in the required interpretation of an illuminated contact: when a contact is illuminated in the fixed support condition, it means that the system "thinks" that it should be selected, regardless of the current selection of the user. In the adaptive support condition, the system only shows the contacts it "thinks" the user should have attention for. It is dependent on whether a contact is selected or not whether the attention is required for possible selection or deselection.

## 4.5. Software architecture

The test environment in which the participants perform the task is developed in Game Maker. The basis for this environment is the implementation used earlier in a preceding experiment within this study. This version is updated to suit the needs of this experiment by removing unnecessary elements and adding the required functionality.

The cognitive model is developed in C#, using the development environment of Microsoft Visual Studio 2005. The communication with the test environment is realized through a TCP/IP connection with a specifically designed protocol.

A Tobii X50 eye-tracker [Tobii Technology 2003] is used to track the gaze of the participants. The cognitive model software can connect to the bundled Tobii Eye Tracker Server to get the gaze data. This connection also uses TCP/IP.

Figure 11 shows the interconnection between all components.

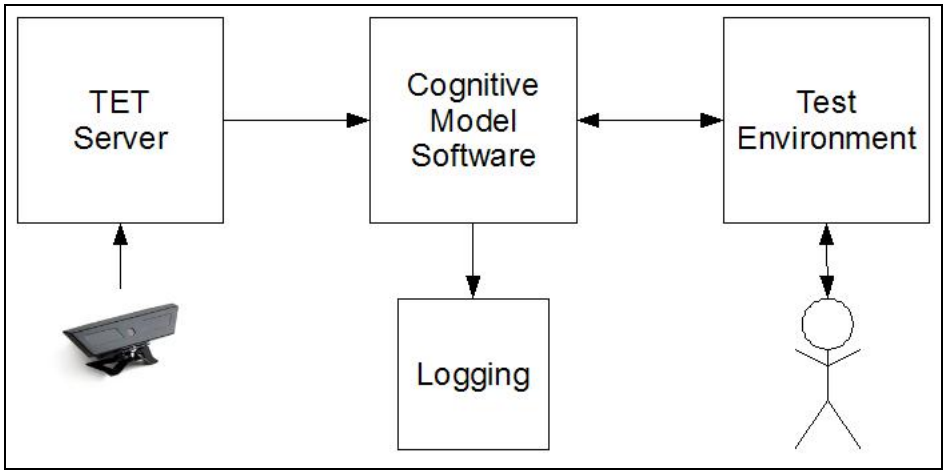



**Figure 11: System Structure**

The above description shows that the various parts of the system all communicate with each other using a TCP/IP network connection. This enables the option to run the model on a different machine than the one where the task is being executed on. Testing has however showed that the available Windows XP workstations are capable to run the test environment, eye-tracker server and cognitive model software on the same machine.

## 4.6. Noise / errors

A very important aspect in the real-life version of the task at hand (TPCT) is the fact that a computer is not able to perform it autonomously perfectly. Several decisive factors in the assessment of the threat level of a contact cannot easily be measured using some type

of sensor. Examples of these factors are cultural or environmental aspects, such as local holidays or weather types which may influence the behavior of contacts.

To replicate this aspect in the experimental setup, some error (noise) has to be added to the determination of the threat levels of the contacts. The system can give an indication of the actual threat level. This indication is however not completely accurate.

### 4.6.1. Requirements

The addition of noise to the threat levels is bounded by some requirements:

1. The noise level should be comparable to the real-life situation.
2. The user task performance in the fixed support condition should increase compared to the no support condition, despite the addition of noise. Otherwise, the addition of support would be useless.
3. The performance of the system should vary between the contacts to avoid predictability.
4. The performance of the system should vary over time to avoid predictability.
5. The performance of the system should increase and decrease gradually to maintain credible to the user.
6. The amount of noise should be comparable for high and low threat levels, since one support system only affects high threat levels, where another system also uses low threat levels. The model performance should not be influenced by a variation of noise between high and low threat levels.
7. The deviation of the noise should be higher than the deviation of the threat values between the contacts. When this deviation would be lower, less rearrangement in the order of contacts when ranked on threat level would occur.
8. Every participant should perform the task with an equal noise level. When using randomization, this would ideally be the same for every run (pre-randomized).
9. The implementation should be as clear as possible. When the design and implementation become more complex, the analysis becomes harder.

All these requirements should be met in the design of the noise addition.

### 4.6.2. Implementation 1: Adding noise

In order to keep the support system credible to the user, the posed mistakes in the support should be reasonable. For instance, when a contact is incorrectly illuminated (wrongly draws the attention), it is better understood and accepted by the user when this is a slight mistake than when the contact is obviously not important in any way. Mistakes can thus not only be expressed by the ratio of correct/incorrect supported contacts, but also by the severity of the error. It is desirable this severity can be controlled directly.

The algorithm in Figure 12 shows a proposed method to add noise to the threat levels of all contacts.

```
maxNoise = 0.1; // maximum deviation from original noise level
maxVariationNoise = 0.01; // maximum deviation per timestamp

foreach Contact c
        if (abs(c.manipulatedThreatLevel - c.threatLevel) < maxNoise)
                c.manipulatedThreatlevel += +/- maxVariationNoise;
        end
end
```

**Figure 12: Algorithm to add noise**

Threat values are manipulated as followed. Threat values always vary between 0 (minimal threat) and 1 (maximal threat). For each of the 25 contacts, a random value between $-\alpha_1$ and $+\alpha_1$ is added to the original value, where x has to be decided through pilot experiments to assure requirement 1, 2, and 7. For now, assume $\alpha_1$ to be 0.1. The order of the contacts, ranked on the manipulated threat level, may now be different from the original order.

Requirement 3 implies that the noise for a contact should change over time to avoid predictability. This can be realized by adding a random value between $\alpha_2$ and $-\alpha_2$ to the original manipulated threat value. Again, these values have to be determined in pilot experiments, but assume $\alpha_2$ to be 0.01 for now. Note that $\alpha_2$ is only added when the sum remains between $\alpha_1$ and $-\alpha_1$.

The last thing to be decided through pilot experiments is the duration of the period between two updates with $\alpha_2$ on the threat values. This time *t* is set on 2000 ms for now.

The above implementation implies that after having added a maximum of +/- $\alpha_1$ at the start, every 2000 ms the threat value for each contact is updated with a maximum of +/- $\alpha_2$.

In order to meet requirement 8, the noise values are only randomized once. After this, $\alpha_1$, manipulated with $\alpha_2$ every 2000 ms are read from a text-file, resulting in the same noise for every run.

### 4.6.3. Implementation 2: Adding false alarms and misses

The implementation described above implies one drawback. During the development of the experiment it became clear that the error rate of the system should be manipulated over time. The above implementation takes care of variation in noise over time, which indirectly affects the number of generated errors. If the average error level of a given period of time is desired to be for example 80%, the α-values can be manipulated such, that this average is reached.

However, the severity of the errors can not be manipulated. The noise implementation implies that the severity of the errors increases, when the error level increases. Severity is an important factor in the addition of errors, since users may react very different to severe errors than to slight errors.

The severity of errors should be constant when the error level changes. In the first implementation, the average severity of the errors rises when the error level is increased. Since the noise is higher, contacts may take bigger jumps in the ranking, creating more

severe errors. To overcome this problem, a second implementation to realize errors is proposed.

Errors in the support can directly be expressed as false alarms and misses. These can also be directly generated. This method is proposed in the algorithm in Figure 13.

```
for i = 1:numberofSwappedContacts
       swap(contacts(random*5), contacts(random*5+5));
end
```

**Figure 13: Algorithm to add false alarms and misses**

A number of contacts (from zero to five) on places in the top five are swapped with places not in the top five. This creates one false alarm and one miss per swap. In order to prevent the errors from being too obvious, the places not in the top five are always in the top ten (thus places five to ten). The accuracy level of the support can now be manipulated from 0-100% in steps of 20%. When a more precise accuracy is required (such as 50%) this can be done by alternating 40% and 60%.

The swaps are pre generated in a random fashion. They are saved into text files to ensure that every run contains the errors on the same moments (between runs and participants). The duration of a swap is ten seconds to prevent a very restless screen.

# 5. Experimental validation

In order to test the hypotheses an experiment was conducted in order to compare the three support conditions with each other and with a no support condition. Before a solid experiment could be designed, pilot experiments were carried out to optimize the setup.

## *5.1. Pilots*

In order to develop a solid, rigorous experiment multiple pilot experiments were performed. This section outlines the motivations for these pilots and the most important observations.

### 5.1.1. Technical issues

The experiment that is needed to test the stated hypotheses is relatively complex. Besides the software that is needed to implement the various types of support models, software is needed in order to log all necessary data and calculate the task performance of each participant on each moment. This software and the software needed to analyze the acquired data afterwards were tested during all the pilots.

Other aspects that were tested during several pilots were the instructions and questionnaires on paper. The participants were always instructed about the fact that the experiment was in a pilot status. Feedback on the understandability and completeness of the paperwork was always asked. One particular pilot was exclusively focused on the questionnaires, without performing the actual task at hand. The participant had performed the task earlier and was asked whether the questions were clear and if they covered all relevant aspects.

### 5.1.2. Learning effect

One of the most important results of the pilot studies was the very strong learning effect. When only one participant performs the task in multiple conditions, the task performance of the second run will always be better than the task performance of the first run, regardless of the given support types. The same applies to the third and second run and so on. An exception is the task performance during the last two runs. A fatigue effect was found here since participants had already performing during four or five conditions of ten minutes each.

The effects of run order on task performance are shown in Figure 14.

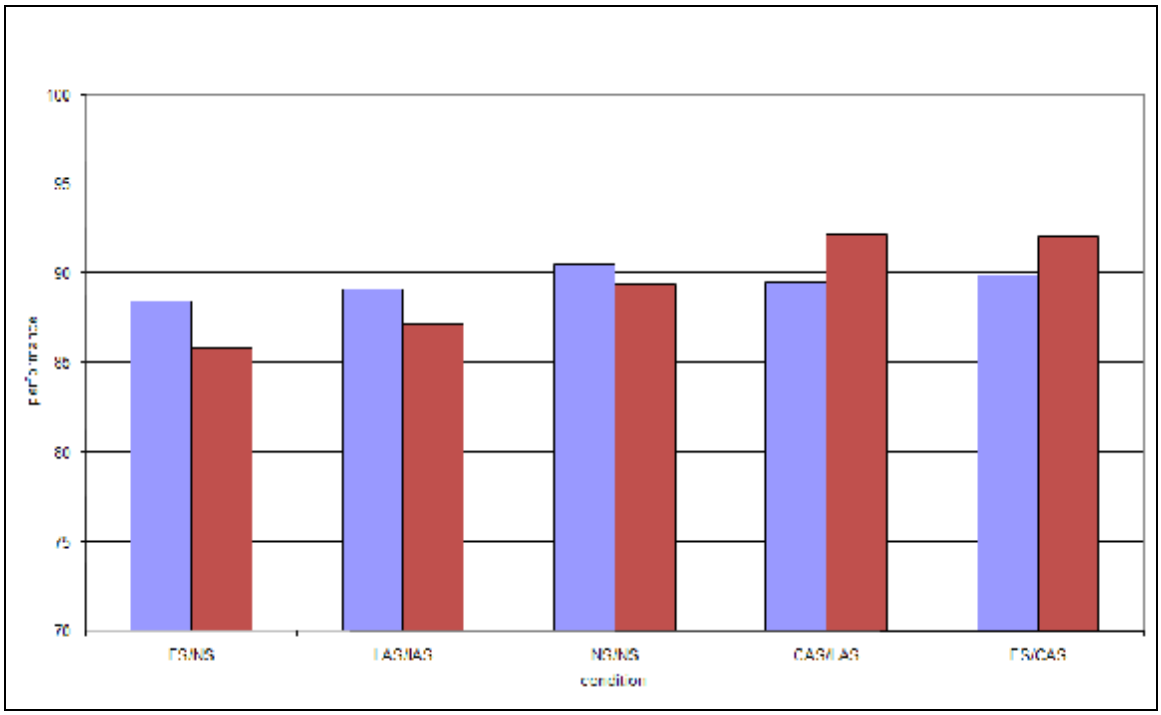



**Figure 14: Task performance during two pilots**

The left blue bars show the runs of one participant, the red right bars show the performance of another. From left to right the order in which the runs were done is presented. We see that despite the fact that the conditions were different for both participants, the task performance increases over time.

The learning effect could be reduced by an increased practice period on beforehand. This practice session is already present, but it is only about three minutes long. The reason to keep the practice session at the same length is the fatigue effect. When the practice session becomes longer, the fatigue effect during the final runs will become stronger.

The conclusion is that it is very hard to predict task performance results during the various support conditions using pilot experiment. The reasons for this are learning effect, fatigue effect and personal differences, such as support preferences and intrinsic task performance.

An attempt to overcome the learning effect during pilot experiments was to make the participant perform in each condition twice, alternating between conditions. For example, when the fixed support condition (FS) was compared to the no support condition (NS), the order in which the conditions are present can be NS-FS-NS-FS. The average of both NS and FS runs can now be compared to each other. The disadvantage of this method is that only two conditions can be compared with one participant, needing a lot more pilot experiments to test multiple conditions. When trying to compare more than two conditions, the pilot experiment would become too long.

In the actual experiment, the order of the conditions can be varied between subjects to balance out the effects of the order.

### 5.1.3. Demonstration of problems with fixed support

In order to demonstrate the advantage of adaptive support over fixed support the problems imposed by the usage of fixed support need to be demonstrated. The anticipated problem is inappropriate reliance. Sections 1.4.2 and 3.1 describe this problem in more detail.

The investigation of the reliance effects was done by varying the support accuracy over time. The task performance of the participant over time should then be reduced during some intervals in the run. A delay in the reaction to for example a dropping support accuracy is expected. The runs in Figure 15 used the task accuracy order *50%-80%-20%-80%-50%* for all conditions.


**Figure 15: Task performance within runs**

We see that no particular part of the runs shows a significant lower task performance than the rest. There is also no significant difference between the first and second half of the interval with the lowest support accuracy (20%).

This method does not work when only using one participant with two runs. Other aspects, such as scenario effects (differences in task performance due to complex/simple parts of the scenario) and personal differences (subtasks of the overall task that particular this participant found hard to do) have caused that the reliance effects could not be shown using pilot experiments. Multiple participants during various sections of the scenario are needed to show the effects.

The pilot experiment did show that the task performance measures that were used up until then were not sensitive enough to give an accurate image of the task performance of the participants. The severity of the made errors and the difficulty of the scenario at a certain moment were not incorporated in the used measures. These measures were the d' score

and the HIT-rate. In the d' score, the hits and false alarms are normalized and then subtracted from each other.

This has lead to a new, more sensitive task performance measure described in section 5.2.5.

## *5.2. Method*

### 5.2.1. Participants

A total of 40 college students (17 male, 23 female) with an average age of 23 years (*SD* 2.6) participated in the experiment as paid volunteers.

### 5.2.2. Task

The task is described in section 4.1. The complex scenario was used, with a duration shortened to ten minutes.

### 5.2.3. Design

A 4 (support type) × 2 (task performance level) design was used. Support type is a within-subjects factor and the order was balanced between the participants. Task performance level was a quasi independent variable we used to categorize participants.

### 5.2.4. Independent variables

The following independent variables are distinguished:

*Support type*

The four conditions are:

1. No Support (NS)
2. Fixed Support (FS)
3. Liberal Adaptive Support (LAS)
4. Conservative Adaptive Support (CAS)

*Task performance level*

After the experiment, a median split was performed to separate good and poor performers in the no support condition. By only looking at this condition (NS) the ability to cope with this particular support type is ruled out. It is expected that poor performers rely more on decision support than good performers. Note that this is a semi-independent variable.

### 5.2.5. Dependent variables

For the scope of this thesis, the only relevant dependent variable is the task performance. Other measures were also taken, such as trust and understandability measures. These are not included in this thesis.

The performance on the tactical picture compilation task was determined by the accuracy of the identification of the five most threatening contacts during the task. The task performance was measured using the following method.

In [Koning et al. 2008], the task performance of the user was measured by looking at the number of hits, misses, correct rejections, and false alarms. The task performance was calculated by subtracting the z-score of the hits from the z-score of the false alarms. The result is the d' value. For a detailed explanation see [Lucassen 2008].

The drawback of this method is that only the number of errors is incorporated in the task performance measure. The severity of the errors is however also an important factor. Two methods to include the severity of errors in the task performance measure are proposed

*Method 1: Ranking the contacts on threat level*

In this method the position in an ordered list (on threat level) of contacts is used as a measure for severity. When for example the participant would not select the 5$^{th}$ most threatening contact but instead of this the 6$^{th}$ most threatening contact, this would not be a severe error. In fact, this is the least significant error which could be made (assuming that there are always 5 selected contacts). However, when the participant does not select the most threatening contact but instead of this selects the least threatening contact, this is a very severe error.

The severity of errors in all contacts based on their relative threat level is shown in Figure 16.

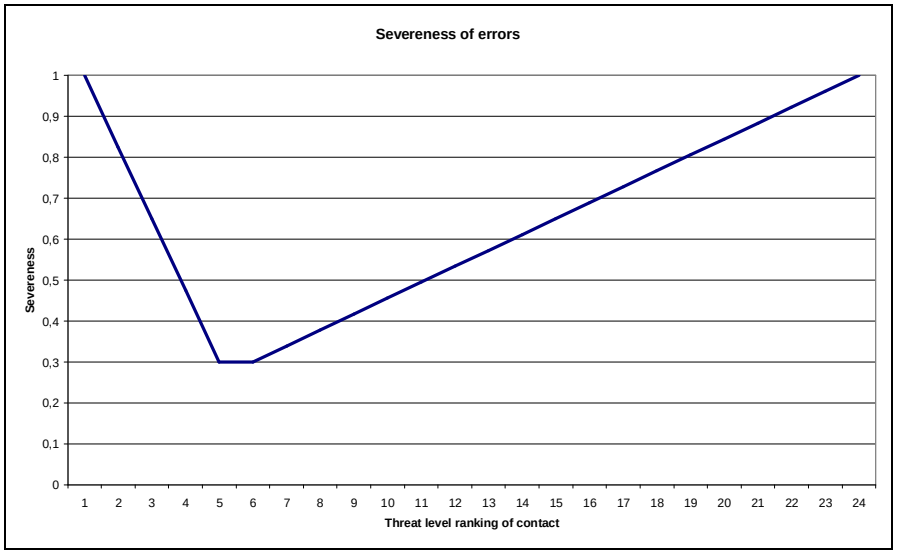



**Figure 16: Severity of errors**

*Method 2: Including deviations in threat levels*

A more accurate method to look at the severity of errors is the threat level of an incorrectly selected contact. For a selected contact which does not belong in the top five, the error is more severe when the difference in threat level between the contact and a contact in the top five is larger. For an unselected contact which does belong in the top five, the error is more severe when the difference in threat level between this contact and a contact not in the top five is larger. The two decisive contacts whether a contact should be selected or not are the number five and six.

The task performance measure is based on penalties for each incorrectly assessed contact. For contacts which should be in the top five (and thus the selection), the penalty is calculated as shown in Figure 17.

```
median = tl[5]+tl[6]/2;

for (i = 1; i < 25; i++)
        if i < 6
                factor = 24/5;
        else
                factor = 24/19;
        end
        pentalty[i] = factor * abs(tl[i]-median);
end
```

**Figure 17: Task performance algorithm**

The *factor* variable denotes the difference between an incorrectly selected contact (thus not in the top five) and an incorrectly unselected contact (thus in the top five). This is the total number of contacts divided by the number of (un)selected contacts.

The penalties are normalized afterwards in order to be able subtract them from the perfect score of 100.

*Learning effect*

The most direct way to compare task performance in these conditions is to make every participant perform the task in these conditions and then look at the task performance scores within subject. However, several pilot studies have shown a very strong learning effect. This results in an increasing task performance over time, no matter what condition is given at that particular time. The influence of the learning effect could be reduced by an extended practice period at the beginning. Unfortunately, the duration of the entire experiment would also increase, which inflicts fatigue effects at the end. These effects result in a lower score for the last sessions.

The effects described above imply that is hard to compare task performance within subjects. This means that it is better to look at task performance between subjects. The learning effect is still present, so we use a Latin square to distribute the order of the conditions.

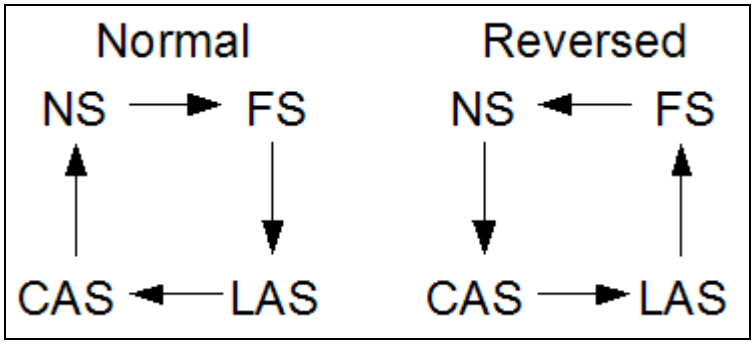



**Figure 18: Latin square with all four conditions**

Figure 18 shows the used Latin square with all conditions. Eight different orders come from this implementation. Appendix B shows the order of conditions for all 40 participants.

### 5.2.6. Procedure

Each experiment has a duration of two hours. The schedule of the experiment is shown in Table 1.

| Activity | Time (minutes) |
|---|---|
| First instructions | 10 |
| Test | 20 |
| Second instructions | 10 |
| Calibration eye-tracker | 5 |
| Practice period | 5 |
| Condition 1 | 10 |
| Break | 5 |
| Condition 2 | 10 |
| Break | 5 |
| Condition 3 | 10 |
| Break | 5 |
| Condition 4 | 10 |
| Questionnaires/interview | 15 |
| Total | 120 |

**Table 1: All activities during the experiment**

During the first instructions, the participant signs the participant contract (see Appendix C) and compensation form. The instructions in Appendix D cover the threat assessment of the contacts. The criteria that are decisive for the assessment are introduced, along with an example.

After this, the participant does a test to check whether the knowledge about the threat assessment is sufficient and to practice with it a little more. The test is completely reviewed together with the participant.

The second set of instructions (see Appendix E) covers the task environment that is used for the experiment. After calibrating the eye-tracker, the participant is given some time to practice with the actual task environment, without any of the support system active (NS). The practice period is under direct supervision of the experimental leader, so the participant can ask questions and the experimental leader can make suggestions for improvement.

Before each condition, the participant is fully instructed about the support (or no support) that is given during the condition. These instructions (see Appendix F) are given as close

to the start of the condition as possible, to make sure that the participant is completely aware of the support functioning.

Between the conditions there is time available for a short break. The participant has to decide for himself whether he is ready for the next condition. After having completed all four conditions the experiment is ended with a questionnaire and a debriefing.

In [Lucassen 2008], the duration of each condition (hard/easy scenario) was about 25 minutes. This quite long duration was necessary to gain enough data on the human attention allocation. This data was only obtained during freezes in the task, where participants selected which contacts they were attending. In this experiment, freezes are no longer present. The only thing that needs to be measured during the conditions is the user task performance. This is a constant measure, which can be calculated for every time span. Keeping the time that a participant needs to get acquainted with the task for each condition in mind, we set the duration of each condition to ten minutes.

During each condition, the performance of the threat assessment of the support is being varied in order to demonstrate the inappropriate reliance. The performance of the support will be varied between 20%, 50%, and 80% accuracy. Three orders are used to rule out the influence of the scenario. Each run is divided into 2-minute blocks with the following accuracies:

1. *50%-80%-20%-80%-50%*
2. *80%-20%-80%-50%-50%*
3. *50%-50%-80%-20%-80%*

The given numbers are the percentages of contacts which are correctly suggested as threatening. In the fixed support condition, these are directly suggested to the user. The adaptive support uses the same threat assessment performance, but with a more advanced algorithm to offer support to the user. Appendix B shows which orders of errors are used in which runs.

The percentages were chosen such that:

1. There is a sufficient gap between the various conditions
2. Some of the support accuracies will be higher than the human task performance and some will be lower. This is tested in section 5.3.6.

## 5.3. Results

This section contains the results of the experiment described in section 5.1 in the light of the original hypotheses from section 1.4.

### 5.3.1. Task performance with fixed support

The first original hypothesis states that the task performance of the participants during the fixed support condition is higher than the task performance without any support.

Figure 19 shows the average task performance for all participants in all conditions.

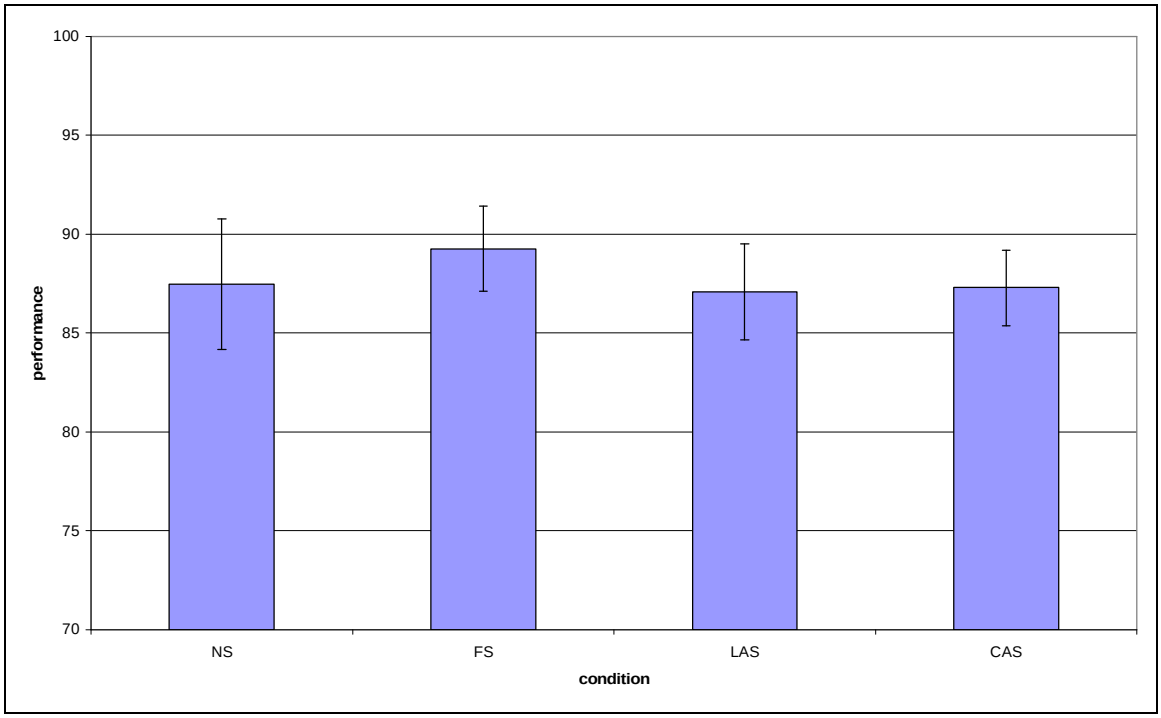



**Figure 19: Average task performance in all conditions**

| Condition | Mean | SD |
|---|---|---|
| NS | 87.5 | 3.3 |
| FS | 89.3 | 2.2 |
| LAS | 87.1 | 2.4 |
| CAS | 87.3 | 1.9 |

**Table 2: Task performance for all conditions**

Table 2 shows the results for all conditions. The task performance with FS is significantly better than all other conditions (NS: $t=2.8$, $p<0.01$, LAS: $t=4.2$, $p<0.01$, CAS: $t=4.3$, $p<0.01$). There is no significant difference between NS, LAS, and CAS.

The first hypothesis can hereby be accepted, since the task performance in the FS condition is significantly higher than in the NS condition. Attention allocation support is useful to improve task performance.

### 5.3.2. Inappropriate reliance on fixed support

To demonstrate the hypothesis that fixed support causes inappropriate reliance, three sub-hypotheses need to be accepted (also see section 1.4.2):

a. Advice is followed, also when the support accuracy is low.

b. Users are sensitive for changes in the support accuracy.

c. Users adapt their behavior to the changes in the support accuracy.

In Figure 19 the task performance of the FS condition is significantly different from the NS condition. Since the only manipulation is the addition of fixed support, the difference in performance implies that the advice is followed and so sub-hypothesis a is accepted.

Figure 20 shows the task performance with the FS, LAS, and CAS support conditions during the intervals with 20%, 50%, and 80% accuracy of the support.

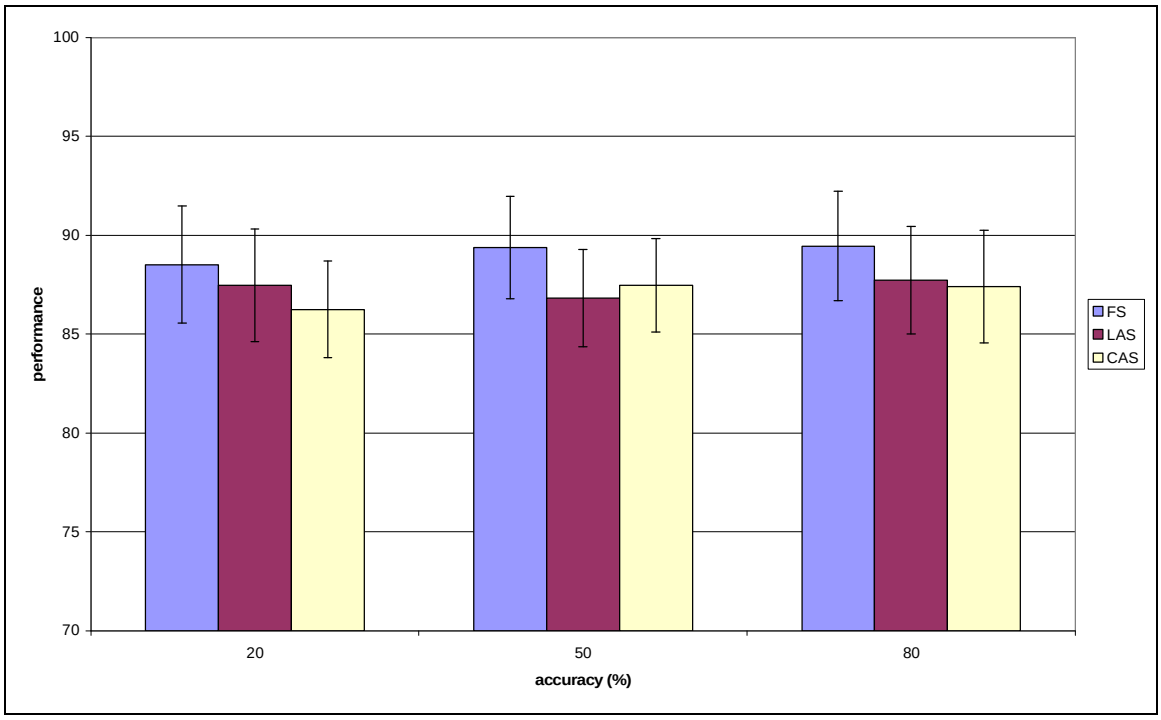



**Figure 20: Task performance during support accuracies**

The task performance in the FS condition increases when the support accuracy rises. This effect is however not significant. For the LAS condition, the differences in task performance are also not significant. For the CAS condition, the increase in task performance is significant for the 20% to 50% condition ($t=2.2$, $p<0.05$) and for the 20% to 80% condition ($t=1.9$,$p<0.05$), but not for the 50% to 80% condition.

Table 3 shows the results during the various support accuracy intervals.

| Condition | 20% accuracy | | 50% accuracy | | 80% accuracy | |
|---|---|---|---|---|---|---|
| | Mean | SD | Mean | SD | Mean | SD |
| FS | 88.5 | 3.0 | 89.4 | 2.6 | 89.5 | 2.8 |
| LAS | 87.5 | 2.9 | 86.8 | 2.5 | 87.7 | 2.7 |
| CAS | 86.3 | 2.4 | 87.5 | 2.4 | 87.4 | 2.9 |

**Table 3: Task performance during 20%, 50%, and 80% support accuracy intervals**

We can see that the influence of the support accuracy is the largest for the FS condition. The difference between performance during the 20%, 50%, and 80% intervals shows that the participants are sensitive to changes in the support accuracy, but the effect is not significant. This means that sub-hypothesis b cannot be accepted.

The difference in task performance during the first and second half of the 20% and 80% support accuracy interval is leading when showing the adaption of the participants on the support. The task performance during NS is subtracted from the task performance during FS to show the effect in Table 4.

| | FS | | LAS | | CAS | |
|---|---|---|---|---|---|---|
| | Mean | SD | Mean | SD | Mean | SD |
| 1st half | 1.4 | 1.7 | 0.8 | 1.7 | -0.7 | 2.0 |
| 2nd half | 1.8 | 1.3 | -0.7 | 2.0 | -1.9 | 1.9 |
| Ratio: | 1.3 | | -0.9 | | 1.9 | |

**Table 4: Task performance with FS-NS during the 20% accuracy intervals**

The second half of the 20% support accuracy interval has a significantly higher task performance ($t=-3.46$, $p<0.01$). This shows that the participants needed some time to adapt their behavior to the altered support accuracy, accepting sub-hypothesis c.

By accepting all three sub-hypotheses, the hypothesis that fixed support causes inappropriate reliance can be accepted. Sub-hypothesis b is however not accepted because the effect is not significant. This means that only a strong suspicion exists that fixed support causes inappropriate reliance.

### 5.3.3. Reduction of inappropriate reliance by adaptive support

Hypothesis 3 states that the usage of adaptive support reduces the inappropriate reliance seen in the fixed support. This can be shown by looking at the same results as in section 5.3.2, but now for both adaptive support conditions.

The first sub-hypothesis states that the advice is followed, also when the support accuracy is low. When looking at Figure 19, we see that both adaptive conditions do not differ significantly from the NS condition. This means that sub-hypothesis a has to be declined.

When looking at the differences in task performance between the various support accuracy levels for the adaptive support conditions, we see that this is not significant for

the LAS condition and only partly significant for the CAS condition (between 20% and 50% support accuracy). On this basis, sub-hypothesis b can also not be accepted.

In both the LAS and CAS condition, the second half of the 20% support accuracy intervals was significantly worse than the first half. This means that the adaption effect is certainly not present, accepting sub-hypothesis c.

Since only sub-hypothesis c is accepted, the hypothesis that adaptive support reduces the inappropriate reliance of the fixed support has to be declined.

### 5.3.4. Task performance for good and poor performers

Hypotheses 4 states that effect of the adaptive support will be bigger for good performers than for poor performers.

Figure 21 shows the average task performance for good and poor performers in all conditions. The distinction between good and poor performers was made by calculating the average task performance in all four runs for each participant. Participants in the top 20 are identified as good performers. The bottom 20 is identified as poor performers.

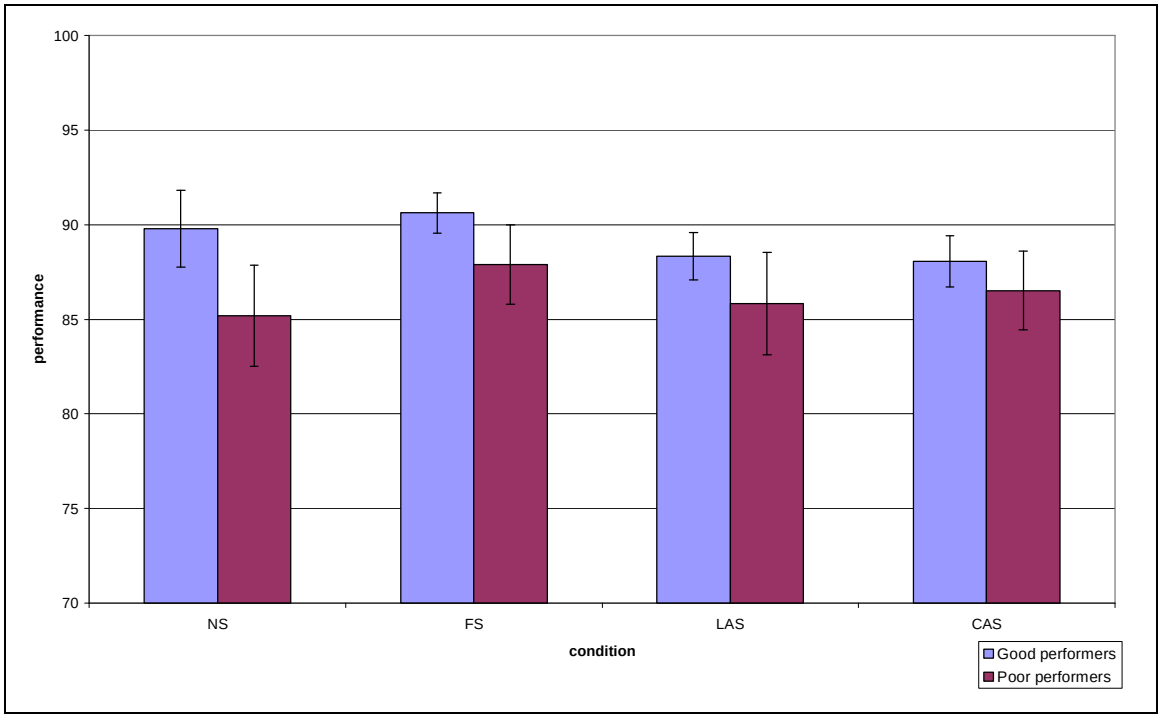



**Figure 21: Task performance for good/poor performers in all conditions**

Table 5 shows the results for good and poor performers in all conditions. In all conditions, the good performers were significantly better than the poor performers (NS: $t=6.1$, $p<0.01$, FS: $t=5.1$, $p<0.01$, LAS: $t=3.8$, $p<0.01$, CAS: $t=2.8$, $p<0.01$).

| Condition | Good performers | | Poor performers | |
|---|---|---|---|---|
| | Mean | SD | Mean | SD |
| NS | 89.8 | 2.0 | 85.2 | 2.7 |
| FS | 90.6 | 1.1 | 87.9 | 2.1 |
| LAS | 88.3 | 1.2 | 85.8 | 2.7 |
| CAS | 88.1 | 1.4 | 86.5 | 2.1 |

**Table 5: Task performance for good/poor performers**

For both good and poor performers, the adaptive support condition did not contribute to the task performance. This declines hypothesis 4.

### 5.3.5. Conservative and liberal setting

Hypothesis 5 states that the task performance during the CAS condition is higher than during the LAS condition. Figure 19 shows that the task performance during the two adaptive support conditions does not significantly differ, which declines this hypothesis.

### 5.3.6. Task performance of the support

This analysis was done to compare the settings of the support accuracy to the human task performance. Note that these last two result sections are not included in the hypotheses.

In order to compare the "task performance" of the support to the task performance of the user, the task performance measure is applied to the manipulation of the support.

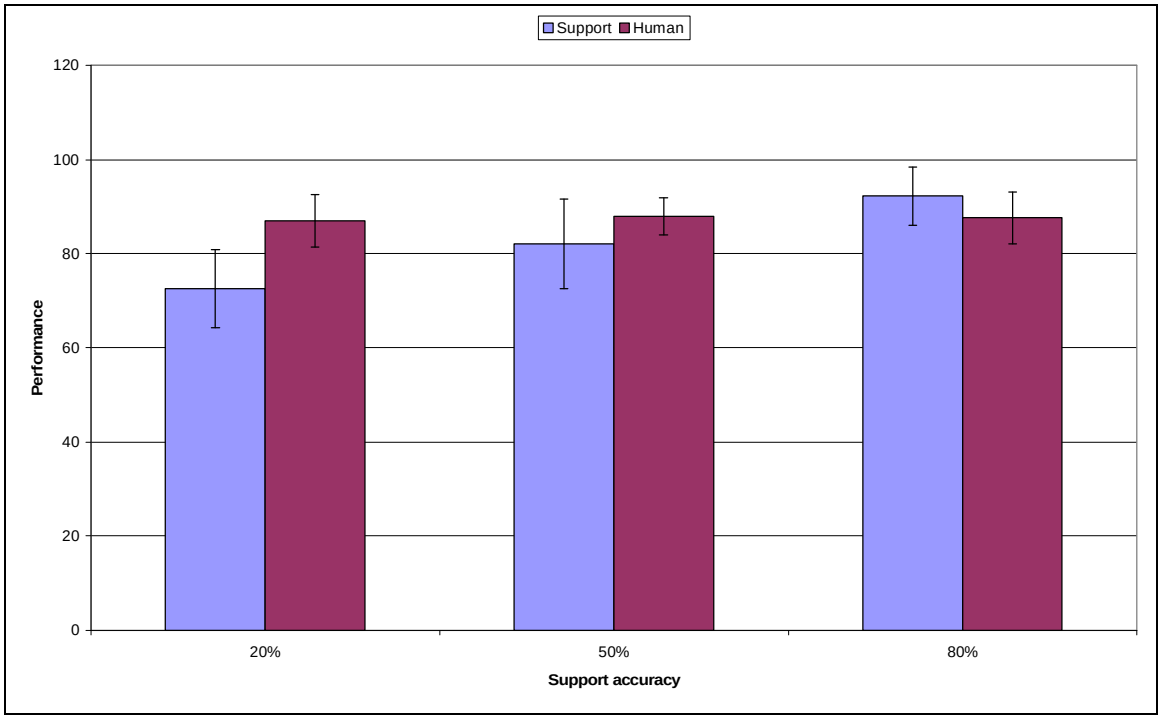



**Figure 22: Task performance of the support with various accuracies**

To compare the support accuracy with human task performance, the bars on the right in Figure 22 illustrate the task performance of the participants without support during the same intervals as this support type was offered.

The task performance of participants without support is between the 50% and 80% support accuracy conditions. This means that during the runs, the support accuracy will vary between a higher and lower performance than the participant.

### 5.3.7. Task performance in runs

A strong learning effect was seen during the pilot studies. As an extra finding, the effects of the order of the runs was also investigated for the final experiment. Note that these last two result sections are not included in the hypotheses.

Figure 23 shows the task performance for all conditions in all runs. All support conditions have occurred on all possible runs. Runs are the first, second, third, and fourth moment where a scenario is started during the experiment.

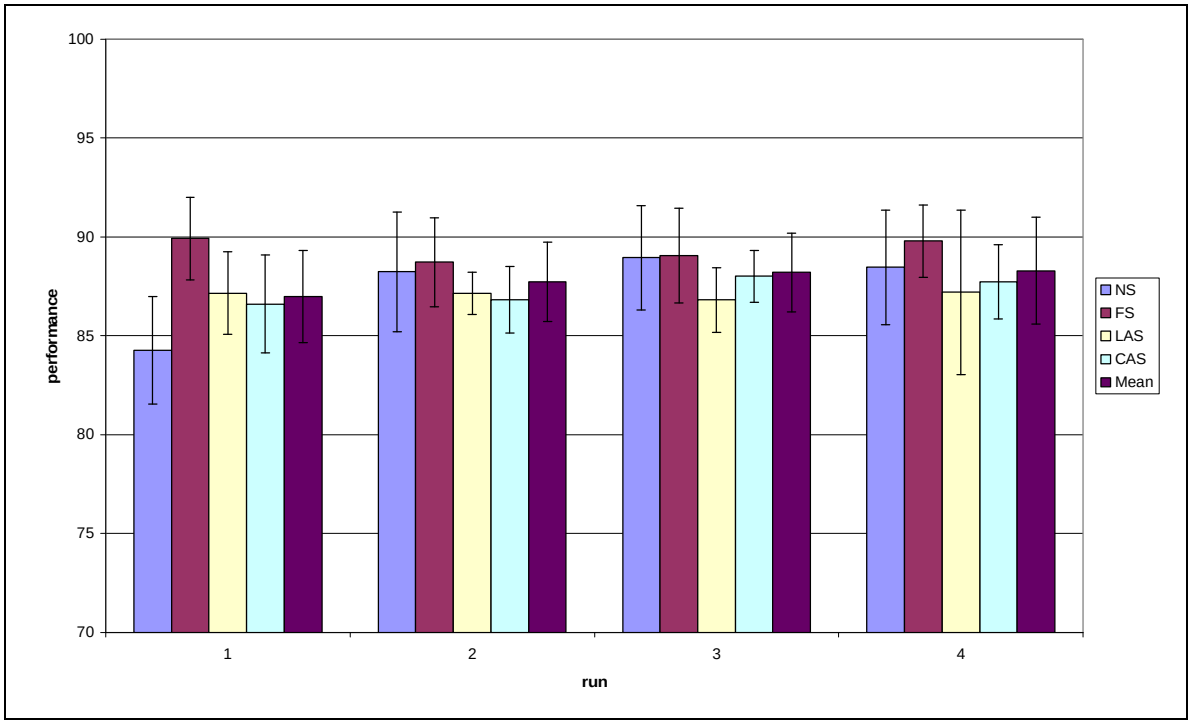



**Figure 23: Task performance in all runs**

Table 6 shows the task performance in all runs. A learning effect can be seen on the average task performance over time. This effect is however not significant between run 1 and 2, run 2 and 3 or run 3 and 4. When looking over a longer period, the effect is significant between run 1 and 4 ($t=2.0$, $p<0.05$). The effect is also significant between run 1 and 3 ($t=1.9$, $p<0.05$), but not between run 2 and 4.

| Condition | Run 1 | | Run 2 | | Run 3 | | Run 4 | |
|---|---|---|---|---|---|---|---|---|
| | Mean | SD | Mean | SD | Mean | SD | Mean | SD |
| NS | 84.3 | 2.7 | 88.2 | 3.0 | 89.0 | 2.6 | 88.5 | 2.9 |
| FS | 89.9 | 2.1 | 88.7 | 2.2 | 89.1 | 2.4 | 89.8 | 1.8 |
| LAS | 87.2 | 2.1 | 87.2 | 1.1 | 86.8 | 1.6 | 87.2 | 4.2 |
| CAS | 86.6 | 2.5 | 86.8 | 1.7 | 88.0 | 1.3 | 87.7 | 1.9 |
| Mean | 87.0 | 2.3 | 87.7 | 2.0 | 88.2 | 2.0 | 88.3 | 2.7 |

**Table 6: Task performance in all runs**

In some cases (e.g. NS, CAS) a slight decrease in task performance can be found between run 4 and run 3. This can be explained by fatigue. After run 4, the participant has cooperated in the experiment for about two hours.

## 5.4. Conclusions

Based on the preceding results, the following conclusions regarding the original hypotheses can be drawn.

The task performance in the fixed support condition is higher than in the no support condition. This means that attention allocation support is useful in this task. The problems with inappropriate reliance are shown in the experiment, although not all results were significant.

When looking at the reliance effects of adaptive support, we have seen that the rise in task performance during the 20% support accuracy interval is not present. In fact, the task performance during the second half of the interval is lower than the first half. However, the overall task performance of adaptive support was not significantly higher than during the no support condition, and the task performance during the various support accuracies are not all significantly different. This means that the hypothesis which states that adaptive support reduces the inappropriate reliance cannot be accepted, although some results were significant.

It was expected that the effect of adaptive support would be bigger for good performers than for poor performers. We have seen that the effect was negative compared to fixed support, so the hypothesis had to be declined. This means that for all participants, the fixed support condition is optimal.

The task performance in the liberal and conservative setting of the adaptive support was not significantly different, which means that the final hypothesis is also not accepted.

The fact that the adaptive support does not yield better results than the fixed support can be caused by multiple factors.

The introduction of adaptivity in support models in this task type could be inappropriate. It is possible that benefits of adaptivity are not bigger than the deficits. The main deficit is the extra complexity inflicted by adaptivity.

The task chosen to apply the support to is the TPCT. It seems that since a well allocated attention is necessary for this task, the support type is appropriate. However, the combination of the task specifics, the support visualization, the support type and the task performance measure could be inappropriate. Given the results of the fixed support condition, it is not likely that the task is not well suited for this support. It is shown that attention allocation support does work in this setting.

Due to availability and cost aspects, no actual marine personnel was used as participants in the experiment. All participants were college students who had one hour training on the threat assessment task. It is likely that the results would be more consistent when expert users are used in the experiment instead of the current novice users. Next to this, a longer duration of the experiment and the addition of more breaks would be desirable. The learning and fatigue effects seen in section 5.3.6 could then also be reduced.

More research could be done on the support accuracy that could be achieved in a realistic scenario. At this point, it is varied between 20%, 50% and 80% accuracy, but it is possible that the accuracy that could be achieved in real-life is quite different from these values. This would harm the external validity of the results.

The task performance measure that was used in this experiment keeps the number of errors into account, along with the severity of the errors by looking at the relative threat level and the difficulty of the current assessment. It is however possible that the factors which are decisive for the task performance do not match the factors on which the participants focus. More research on this match could be done in order to optimize the task performance measure.

Since the task performance did not significantly improve with adaptive support, it is hard to draw conclusions from the results on good and poor performers and the liberal and conservative setting. However, the inappropriate reliance effects seen in the fixed support condition did not show at both adaptive support conditions. This is a strong motivation for further research on this topic. The main question remains: How can adaptivity be used to improve task performance and reduce reliance effects?

# 6. Bibliography


Bailey, B.P., and Konstan, J.A. 2006. On the need for attention-aware systems: Measuring effects of interruption on task performance, error rate and affective state. *Computers in Human Behavior 22 (2006) 685-708.*

Beilock, S.L., Carr, T.H., MacMahon, C., and Starkes, J.L. 2002. When paying attention becomes counterproductive: Impact of divided versus skill-focused attention on novice and experienced performance of sensorimotor skills. *Journal of Experimental Psychology: Applied. Vol 8(1), Mar 2002, 6-16.*

Bolia, R.S., D'Angelo, W.R., and McKinley, R.L. 1999. Aurally Aided Visual Search in Three-Dimensional Space. *Human Factors, 41 (4), 664-669.*

Conner, C. E., Egeth, H. E., and Yantis, S. 2004. Visual Attention: Bottom-up Versus Dispatch Top-Down. *Current Biology, 14, R850-R852.*

Dekker, S.W.A., and Woods, D.D. 2002. MABA-MABA of Abracadabra? Progress on Human-Automation Co-ordination. *Cognition, Technology & Work, 4, 240-244.*

Di Nocera, F., Camilli, M., and Terenzi, M. 2006. Using the distribution of eye fixations to assess pilots' mental workload. *Proceedings of the Human Factors and Ergonomics Society's 50th Annual Meeting*.

Dzindolet, M. T., Peterson, S. A., Pomranky, R. A., Pierce, L. G., and Beck, H. P. 2003. The role of trust in automation reliance. *International Journal of Human-Computer studies, 58, 697-718.*

Endsley, M.R., and Kiris, E.O. 1995. The Out-of-the-Loop Performance problem and Level of Control in Automation. *Human Factors, 37(2), 381-394.*

Fisher, B., Dill, J., and Liljefors, M. 1999. Perceptula cognition and the design of air traffic control interfaces. *Unknown.*

Galster, S. M., and Parasuraman, R. (2004). Task dependencies in stage-based examinations of the effects of unreliable automation. *In D. A. Vincenzi, M. Mouloua, & P. A. Hancock (Eds.), Human performance, situation awareness, and automation: Current research and trends, Vol. II (pp. 23-27). Mahwah, NJ: Lawrence Erlbaum Associates, Inc.*

Gibson, J. J. 1974. Overt and covert attention. *Cornell University, unpublished. http://huwi.org/gibson/overt.php.*

Grootjen, M., Bierman, E.P.B., and Neerincx, M.A. 2006. Optimizing cognitive task load in naval ship control centres: Design of an adaptive interface. *IEA 2006: 16th World Congress on Ergonomics.*

Harris, W.C., Hancock, P.A., and Arthur, E.J. 1993. The effect of taskload projection on automation use, performance and workload. *Proceedings of the 7th International Symposium on Aviation Psychology.*

Heuvelink, A. 2006. Modelling cognition as querying a database of labeled beliefs. *Proceedings of the 7th International Conference on Cognitive Modelling, 365-366.*

Hilburn, B., Jorna, P. G., Byrne, E. A., and Parasuraman, R. 1997. The effect of adaptive air traffic control (ATC) decision aiding on controller mental workload. *In M. Mouloua & J. Koonce (Eds.), Human–automation interaction: Research and practice 84–91. Mahwah, NJ: Erlbaum.*

Horrey, W., and Wickens, C. D. Supporting situation assessment through attention guidance: A cost-benefit and depth of processing analysis. *Proceedings of the 45th Annual Meeting of the Huma Factors and Ergonomics Society.*

Inagaki, T. 2003. Adaptive Automation: Sharing and Trading of Control. *In E. Hollnagel (Ed.), Handbook of Cognitive Task Design (147-169). Mahwah, NJ: Lawrence Erlbaum Associates.*

Itoh, H., Mitomo, N., Matsuoka, T., and Murohara, Y. 2004. An extension of the M-Shel model for analysis of human factors at ship operation. *3rd International Conference on Collision and Groundings of Ships, 118-122.*

Jonides, J., and Yantis, S. 1988. Uniqueness of abrupt visual onset in capturing attention. *Perception & Psychophysics 1988, 43 (4), 346-354.*

Kahneman, D. 1973. Attention and effort. *Prentice Hall, Englewoods Cliffs, NJ.*

Koning, L. de, Maanen, P.-P. van, and Dongen, K. van 2008. Effects of Task Performance and Task Complexity on the Validity of Computational Models of Attention. *Proceedings of the Human Factors and Ergonomics Society's 52nd Annual Meeting*.

Lee, J., and Sanquist, T. 2000. Augmenting the operator function model with cognitive operations: Assessing the cognitive demands of technological innovation in ship navigation. *IEEE transactions on systems, man and cybernetics – part A: Systems and humans 30, 3.*

Lucassen, T. 2008. Cognitive Models of Attention. *TNO-report TNO-DV 2008 S218*.

Mack, A., and Rock, I. 1998. Inattentional blindness. *MIT Press 1998.*

McCarley, J.S., Wickens, C.D., Goh, J., and Horrey, W.J. 2002. A computational model of attention/situation awareness. *Proceedings of the Human Factors and Ergonomics Society 46th Annual Meeting.*

McClernon, C.K., Kaber, D.B., Perry, C.M., and Segall, N. 2006. Towards a sensitive measure of situation awareness in adaptively automated systems. *Proceedings of the Human Factors and Ergonomics Society 50th Annual Meeting 275-279.*

Parasuraman, R., and Riley, V. 1997. Humans and Automation: Use, Misuse, Disuse, and Abuse. *Human Factors, 39(2), 230-253.*

Pavel, M., Wang, G., and Li, K. 2003. Augmented Cognition: Allocation of Attention. *Proceedings of the 36th Annual Hawaii International Conference on System Sciences.*

Psaraftis, H., Caridis, P., Desypris, N., Panagakos, G., and Ventikos, N. 1998. The human element as a factor in marine accidents. *IMLA -10 Conference.*

Pylyshyn, Z. W. 2001. Visual indexes, preconceptual objects, and situated vision. *Cognition 80, 1, 127-158.*

Pylyshyn, Z. W., and Annan, V. J. 2006. Dynamics of target selection in multiple object tracking (MOT). *Spatial Vision 19, 485-504.*

Remington, R.W., Folk, C.L., and McLean, J.P. 2001. Contingent attentional capture or delayed allocation of attention? *Perception & Psychonomics 2001, 63 (2), 298-307.*

Roda, C., and Thomas, J. 2005. Attention Aware Systems. *Encyclopaedia of HCI.*

Rovira, E., McGarry, K., and Parasuraman, R. 2002(I). Effects of information and decision automation on multi-task performance. *Proceedings of the Human Factors and Ergonomics Society 46th Annual Meeting 327-331.*

Rovira, E., McGarry, K., and Parasuraman, R. 2002(II). Effects of unreliable automation on decision making in command and control. *Proceedings of the Human Factors and Ergonomics Society 46th Annual Meeting 428-432.*

Santoro, T., and Thomas, J. 2005. A comparison of air defense warfare task performance with and without an automated task manager using a goms modeling tool. *10th internation command and control research and technology symposium, the future of C2.*

Sarter, N.B. 2000. The need for multisensory interfaces in support effective attention allocation in highly dynamic event-driven domains: The case of cockpit automation. *The International Journal of Aviation Psychology, 10 (3), 231-245.*

Sklar, A.E., and Sarter, N.B. 1999. Good vibrations: Tactile feedback in support of attention allocation and human-automation coordination in event-driven domains. *Human Factors, 41 (4), 543-552.*

Taylor, R. 2001. Cognitive cockpit systems engineering: Pilot authorization and control of tasks. *Defence Science and Technology Laboratory, Human Sciences.*

Taylor, R., Brown, C., and Dickson, B. 2002. From safety net to augmented cognition: Using flexible autonomy levels for on-line cognitive assistance and automation. *NATO RTO HFM symposium on Spatial Disorientation in Military Vehicles: Causes, Consequences and Cures.*

Theeuwes, J., and Chen, C.Y.D. 2005. Attentional capture and inhibition (of return): The effect on perceptual sensitivity. *Perception & Psychophysics 2005, 67 (8), 1305-1312.*

Tobii Technology AB 2003. Tobii 50 Series product description. *http://www.bunnyfoot.com/services/Product_Description_Tobii_50_Series.pdf*

Wickens, C. D. 1984. Processing resources in attention. *In Parasuraman & D. R. Davies (Eds.), Varieties of attention, 63–102.*

Wickens, C. D., and Hollands, J.G. 2000. Engineering psychology and human performance (3rd ed.). *Englewood Cliffs, NJ- Prentice-Hall.*

Wickens, C. D., Helleberg, J., Goh, J., Xu, X., and Horrey, W. 2001. Pilot task management: Testing an attentional expected value model of visual scanning. *Tech. rep. ARL-01-14/NASA-01-7, Aviation Research Lab, Institute of Aviation.*

Wickens, C. D., and McCarley, J. S. 2007. Applied Attention Theory. *Lawrence Erlbaum Associates Inc, Us, chapter 3.*

# Appendix A: Source code

This appendix contains the source code of the *Action* module which was added to the support system to create the adaptive support. Note that this code only contains the liberal variant. Information about the focus of attention of the user is added later. The outcome of the algorithm below is only given as support when the attention for the supported contact is below a threshold value.

```
using System;
using System.IO;
using System.Collections;
using System.Collections.Generic;
using System.Text;

namespace attentionmodel
{
    /// <summary>
    /// This class was written by Teun Lucassen (2008).
    /// Nothing of this code may be used or copied without the permission of the author.
    ///
    /// This software automatically detects faulty allocation of attention to objects and
manipulates it in order
    /// to reduce errors.
    /// </summary>
    ///

    class Action
    {
        double alpha1;                    // Percentage of selected contacts eligible for
support (set in model)
        double alpha2;                   // Percentage of unselected contacts eligible for
support (set in model)

        bool loadNoise;                   // set true when noise is loaded from file, for
new noise set to false (set in model)
        bool saveNoise;                   // set true when you want to save the new noise
(set in model)
        double maxNoise;                    // maximum noise level for all contacts, only
used with new noise (set in model)
        double variationNoise;            // maximum change in noise per cycle, only used
with new noise (set in model)
        int noiseInterval;               // duration of a noise cycle in ms
        double[,] noiseValues;           // noise values for all contacts
        long lastTimeStamp = 0;          // last timestamp for noise cycle
        int t = -1;                      // index of current noise value in array

        public double[] supportValues;  // Luminance values for the adaptive manipulation

        // ERROR MANIPULATION VALUES
        int interval = 10;               // interval for error change = 10 seconds
        int levelInterval = 12;          // duration of each level of error rates
        int numberOfLevels = 5;          // 5 levels of errorrates
        int[] errorLevels = new int[5]; // the amount of incorrect supported contacts in
the top 5 for each
        int[,] errors;                   // actual errors
        int errorFileNumber;             // number of the error file
        Random r;

        public   Action(double   alpha1,   double   alpha2,   double   maxNoise,   double
variationNoise, bool loadNoise, bool saveNoise, int noiseInterval, int errorFileNumber)
        {
            r = new Random();

            this.alpha1 = alpha1;
            this.alpha2 = alpha2;
            this.maxNoise = maxNoise;
            this.variationNoise = variationNoise;
            this.noiseInterval = noiseInterval;
```

```
            this.loadNoise = loadNoise;
            this.saveNoise = saveNoise;
            this.errorFileNumber = errorFileNumber;

            this.supportValues = new double[25];

            errorLevels[0] = 2; // 60%
            errorLevels[1] = 1; // 80%
            errorLevels[2] = 3; // 40%
            errorLevels[3] = 1; // 80%
            errorLevels[4] = 2; // 60%

            errors = new int[5, numberOfLevels * levelInterval];

            if (loadNoise)
            {
                LoadErrors();
            }
            else
            {
                InitializeErrors();
                if (saveNoise)
                    SaveErrors();
            }
        }

        public void UpdateContacts(Stack contacts, double[] selectedThreatValues,
double[] unselectedThreatValues)
        {
            //determine number of selected/unselected contacts
            int numberSelected = 0;
            int numberUnselected = 0;
            foreach (GameMakerContact contact in contacts)
            {
                if (contact.status == 1)
                {
                    numberSelected++;
                }
                else
                {
                    numberUnselected++;
                }
            }

            //determine number of supported contacts
            int numberSelectedSupport = (int)(alpha1 * numberSelected);
            int numberUnselectedSupport = (int)(alpha2 * numberUnselected);

            //reset supportValues
            this.supportValues = new double[selectedThreatValues.Length];

            // set support for selected contacts
            for (int i = 0; i < numberSelectedSupport; i++)
            {
                int minID = Min(selectedThreatValues);
                supportValues[minID] = 1.0; //1 can be replaced by DP information later
                selectedThreatValues[minID] = 2.0; //higher than max
            }

            // set support for unselected contacts
            for (int i = 0; i < numberUnselectedSupport; i++)
            {
                int maxID = Max(unselectedThreatValues);
                supportValues[maxID] = 1.0; //1 can be replaced by DP information later
                unselectedThreatValues[maxID] = -1.0; //higher than max
            }
        }

        public void InitializeErrors()
        {
            for (int i = 0; i < numberOfLevels; i++) // for each error level
```

```
            {
                for (int j = 0; j < levelInterval; j++) // for the duration of the error
level
                {
                    List<int> usedContacts = new List<int>();
                    for (int k = 0; k < errorLevels[i]; k++) // for each error in the
error level
                    {
                        int contactInTop5 = GetRandomContact();
                        int contactOutsideTop5 = 5 + GetRandomContact();

                        while (usedContacts.Contains(contactInTop5))
                        {
                            contactInTop5 = GetRandomContact();
                        }

                        while (usedContacts.Contains(contactOutsideTop5))
                        {
                            contactOutsideTop5 = 5 + GetRandomContact();
                        }

                        usedContacts.Add(contactInTop5);
                        usedContacts.Add(contactOutsideTop5);

                        errors[contactInTop5 - 1, i * levelInterval + j] =
contactOutsideTop5;
                    }
                }
            }
        }

        public int GetRandomContact()
        {
            double d = r.NextDouble();
            int contact = 0;
            if (d < 0.2)
                contact = 1;
            else if (d < 0.4)
                contact = 2;
            else if (d < 0.6)
                contact = 3;
            else if (d < 0.8)
                contact = 4;
            else
                contact = 5;

            return contact;
        }

        public double[] AddErrors(double[] threatValues)
        {
            long currentTimeStamp = DateTime.Now.Ticks;

            if (currentTimeStamp > lastTimeStamp + 10000000 * interval || lastTimeStamp
== 0)
            {
                t = (t + 1) % errors.GetLength(1);
                lastTimeStamp = currentTimeStamp;
            }

            for (int i = 0; i < errors.GetLength(0); i++)
            {
                if (errors[i,t] != 0)
                {
                    double tempThreatValue = threatValues[i]; //threatvalue of contact op
in top 5
                    threatValues[i] = threatValues[errors[i,t]-1];
                    threatValues[errors[i,t]-1] = tempThreatValue;
                }
            }
            return threatValues;
```

```
        }

        public double[] AddNoise(double[] threatValues)
        {
            double[] resultValues = threatValues;
            long currentTimeStamp = DateTime.Now.Ticks;

            // if cycle has ended, go to next noise value
            if (currentTimeStamp > lastTimeStamp + 10000 * noiseInterval || lastTimeStamp
== 0)
            {
                t = (t + 1) % noiseValues.GetLength(1);
                lastTimeStamp = currentTimeStamp;
            }

            // add noise values to threat values
            for (int i = 0; i < threatValues.Length; i++)
            {
                if (threatValues[i] != 2.0 && threatValues[i] != -1.0)
                    resultValues[i] = threatValues[i] + noiseValues[i, t];
                if (threatValues[i] < 0)
                    resultValues[i] = 0;
                if (threatValues[i] > 1)
                    resultValues[i] = 1;
            }

            return resultValues;
        }

        //initialize noise values
        private void InitializeNoiseValues()
        {
            Random r = new Random();
            for (int i = 0; i < noiseValues.GetLength(0); i++) //initialize noise at t=0;
            {
                noiseValues[i, 0] = 0 - maxNoise + 2 * r.NextDouble() * maxNoise;
            }

            for (int i = 1; i < noiseValues.GetLength(1); i++)
            {
                for (int j = 0; j < noiseValues.GetLength(0); j++) // initialize noise at
t>0
                {
                    if (r.NextDouble() > 0.5) // add noise
                    {
                        if (noiseValues[j, i-1] + variationNoise < maxNoise) // maximum
not reached
                        {
                            noiseValues[j, i] = noiseValues[j, i - 1] + r.NextDouble() *
variationNoise;
                        }
                        else
                        {
                            noiseValues[j, i] = noiseValues[j, i - 1] - r.NextDouble() *
variationNoise;
                        }
                    }
                    else // subtract noise
                    {
                        if (noiseValues[j, i-1] - variationNoise > 0 - maxNoise)
                        {
                            noiseValues[j, i] = noiseValues[j, i - 1] - r.NextDouble() *
variationNoise;
                        }
                        else
                        {
                            noiseValues[j, i] = noiseValues[j, i - 1] + r.NextDouble() *
variationNoise;
                        }
                    }
                }
```

```
    }
}

public void LoadErrors()
{
    TextReader tr = new StreamReader("errors" + errorFileNumber + ".txt");



    String line;
    for (int i = 0; (line = tr.ReadLine()) != null; i++)
    {
        string[] values = line.Split(' ');
        for (int j = 0; j < values.Length - 1; j++)
        {
            if (values[j] != "")
                errors[j, i] = Convert.ToInt32(values[j]);
        }
    }
    tr.Close();
}

//load noise values from file
private void LoadNoiseValues()
{
    TextReader tr = new StreamReader("noise.txt");

    String line;
    for (int i = 0; (line = tr.ReadLine()) != null; i++ )
    {
        string[] values = line.Split(' ');
        for (int j = 0; j < values.Length-1; j++)
        {
            if (values[j] != "")
                noiseValues[j, i] = Convert.ToDouble(values[j]);
        }
    }
    tr.Close();
}

public void SaveErrors()
{
    TextWriter tw = new StreamWriter("errors" + errorFileNumber + ".txt");

    for (int i = 0; i < errors.GetLength(1); i++)
    {
        string output = "";
        for (int j = 0; j < errors.GetLength(0); j++)
        {
            output = output + errors[j, i] + " ";
        }

        tw.WriteLine(output);
    }

    // close the stream
    tw.Close();
}

//save noise values to file
public void SaveNoiseValues()
{
    TextWriter tw = new StreamWriter("noise.txt");

    for (int i = 0; i < noiseValues.GetLength(1); i++)
    {
        string output = "";
        for (int j = 0; j < noiseValues.GetLength(0); j++)
        {
            output = output + noiseValues[j, i] + " ";
        }
```

```
                tw.WriteLine(output);
            }

            // close the stream
            tw.Close();
        }

        // Find maximum in Array a
        private int Max(double[] a)
        {
            double max = a[0];
            int maxID = 0;

            for (int i = 1; i < a.Length; i++)
            {
                if (a[i] > max)
                {
                    max = a[i];
                    maxID = i;
                }
            }
            return maxID;
        }

        // Find minimum in Array a
        private int Min(double[] a)
        {
            double min = a[0];
            int minID = 0;

            for (int i = 1; i < a.Length; i++)
            {
                if (a[i] < min)
                {
                    min = a[i];
                    minID = i;
                }
            }
            return minID;
        }
    }
}
```

# Appendix B: Order of conditions for all participants

This appendix shows the orders in which the participants received the four conditions. The "error" column denotes the error file used in each run.

| # | run 1 | | run 2 | | run 3 | | run 4 | |
|---|---|---|---|---|---|---|---|---|
| | condition | error | condition | error | condition | error | condition | error |
| 1 | NS | 0 | FS | 1 | LAS | 2 | CAS | 3 |
| 2 | FS | 1 | LAS | 2 | CAS | 3 | NS | 0 |
| 3 | LAS | 1 | CAS | 2 | NS | 0 | FS | 3 |
| 4 | CAS | 1 | NS | 0 | FS | 2 | LAS | 3 |
| 5 | NS | 0 | CAS | 1 | LAS | 2 | FS | 3 |
| 6 | CAS | 1 | LAS | 2 | FS | 3 | NS | 0 |
| 7 | LAS | 1 | FS | 2 | NS | 0 | CAS | 3 |
| 8 | FS | 1 | NS | 0 | CAS | 2 | LAS | 3 |
| 9 | NS | 0 | FS | 1 | LAS | 2 | CAS | 3 |
| 10 | FS | 1 | LAS | 2 | CAS | 3 | NS | 0 |
| 11 | LAS | 1 | CAS | 2 | NS | 0 | FS | 3 |
| 12 | CAS | 1 | NS | 0 | FS | 2 | LAS | 3 |
| 13 | NS | 0 | CAS | 1 | LAS | 2 | FS | 3 |
| 14 | CAS | 1 | LAS | 2 | FS | 3 | NS | 0 |
| 15 | LAS | 1 | FS | 2 | NS | 0 | CAS | 3 |
| 16 | FS | 1 | NS | 0 | CAS | 2 | LAS | 3 |
| 17 | NS | 0 | FS | 1 | LAS | 2 | CAS | 3 |
| 18 | FS | 1 | LAS | 2 | CAS | 3 | NS | 0 |
| 19 | LAS | 1 | CAS | 2 | NS | 0 | FS | 3 |
| 20 | CAS | 1 | NS | 0 | FS | 2 | LAS | 3 |
| 21 | NS | 0 | CAS | 1 | LAS | 2 | FS | 3 |
| 22 | CAS | 1 | LAS | 2 | FS | 3 | NS | 0 |
| 23 | LAS | 1 | FS | 2 | NS | 0 | CAS | 3 |
| 24 | FS | 1 | NS | 0 | CAS | 2 | LAS | 3 |
| 25 | NS | 0 | FS | 1 | LAS | 2 | CAS | 3 |
| 26 | FS | 1 | LAS | 2 | CAS | 3 | NS | 0 |
| 27 | LAS | 1 | CAS | 2 | NS | 0 | FS | 3 |
| 28 | CAS | 1 | NS | 0 | FS | 2 | LAS | 3 |
| 29 | NS | 0 | CAS | 1 | LAS | 2 | FS | 3 |
| 30 | CAS | 1 | LAS | 2 | FS | 3 | NS | 0 |
| 31 | LAS | 1 | FS | 2 | NS | 0 | CAS | 3 |
| 32 | FS | 1 | NS | 0 | CAS | 2 | LAS | 3 |
| 33 | NS | 0 | FS | 1 | LAS | 2 | CAS | 3 |
| 34 | FS | 1 | LAS | 2 | CAS | 3 | NS | 0 |
| 35 | LAS | 1 | CAS | 2 | NS | 0 | FS | 3 |
| 36 | CAS | 1 | NS | 0 | FS | 2 | LAS | 3 |
| 37 | NS | 0 | CAS | 1 | LAS | 2 | FS | 3 |
| 38 | CAS | 1 | LAS | 2 | FS | 3 | NS | 0 |
| 39 | LAS | 1 | FS | 2 | NS | 0 | CAS | 3 |
| 40 | FS | 1 | NS | 0 | CAS | 2 | LAS | 3 |

# Appendix C: Participant contract

**Proefpersoonverklaring**

Projectnummer: 013.65063

Ondergetekende,

Naam ..... m / v

Adres en woonplaats .....

……..

Geboortedatum .....

verklaart op vrijwillige basis deel te nemen aan het experiment “Ondersteuning door een sociale computer” bij TNO Defensie en Veiligheid te Soesterberg. Het is mij duidelijk dat het daarbij gaat om een onderzoek naar het ondersteunen van aandachtsallocatie.

De bedoelingen van het experiment en de daarbij gevolgde aanpak zijn tot mijn tevredenheid uitgelegd, en mijn vragen zijn door de proefleider helder beantwoord. Er is mij verzekerd dat ik op elk moment zonder opgaaf van redenen mijn deelname aan het experiment kan beëindigen zonder dat dit voor mij nadelige consequenties heeft. Evenzo kan de proefleider onder dezelfde voorwaarden mijn deelname aan het experiment beëindigen.

Bij rapportage van de resultaten van het experiment wordt mijn privacy beschermd in die zin dat het niet mogelijk zal zijn op enigerlei wijze mijn identiteit te achterhalen. Een uitzondering hierop kan worden gevormd door de presentatie van dia-, foto- of videomateriaal, echter alleen nadat ik daarvoor expliciet mijn toestemming heb gegeven.

Voorts verklaar ik lichamelijk in goede gezondheid te verkeren.

Soesterberg, .. - .. - 2008

Handtekening proefpersoon: Paraaf proefleider:

# Appendix D: Participant instructions on threat levels

## *Opbouw van het experiment*

### Inleiding

U bent officier van de wacht op een marine schip en uw taak is om van de schepen (deze worden contacten genoemd) die in de omgeving varen te bepalen of deze een bedreiging vormen voor uw eigen schip of niet (dat heet beeldopbouw). Van alle contacten moet u steeds de 5 meest bedreigende contacten identificeren. Het bepalen of een contact bedreigend is, gebeurt op basis van een aantal criteria. Voordat het eigenlijke experiment begint krijgt u eerst een toets om te oefenen met de criteria. Deze toets is bedoeld om te kijken of de criteria duidelijk zijn. Na de toets worden de antwoorden en eventuele onduidelijkheden besproken. Hierna krijgt u de gelegenheid om de taak kort te oefenen en is er nog ruimte voor vragen alvorens het experiment begint.

Opbouw:

- Uitleg criteria
- Toets
- Uitleg experiment
- Oefenscenario
- Uitleg ondersteuning
- Deel 1 (10 min)
- Deel 2 (10 min)
- Deel 3 (10 min)
- Deel 4 (10 min)
- Afsluitende questionnaire

In totaal zal het experiment ongeveer 2 uur duren.

### Uitleg van de criteria

Op een radarscherm zijn het eigen schip en de verschillende contacten te zien. Het eigen schip ligt stil. De contacten verplaatsen zich wel en verschillen van elkaar op een aantal criteria. Het bepalen of een contact bedreigend is, gebeurt op basis van de volgende criteria:

- De **koers** die de contacten varen.
  Een contact dat naar uw schip toe vaart is dreigender dan een contact dat van u af vaart.

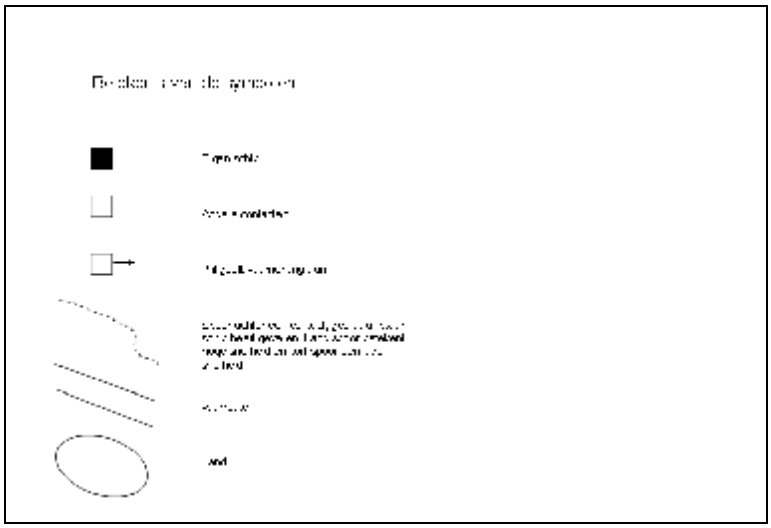
- De **afstand** tussen eigen schip en andere contacten.
  Hoe kleiner de afstand tussen u en een ander contact hoe dreigender een contact is.

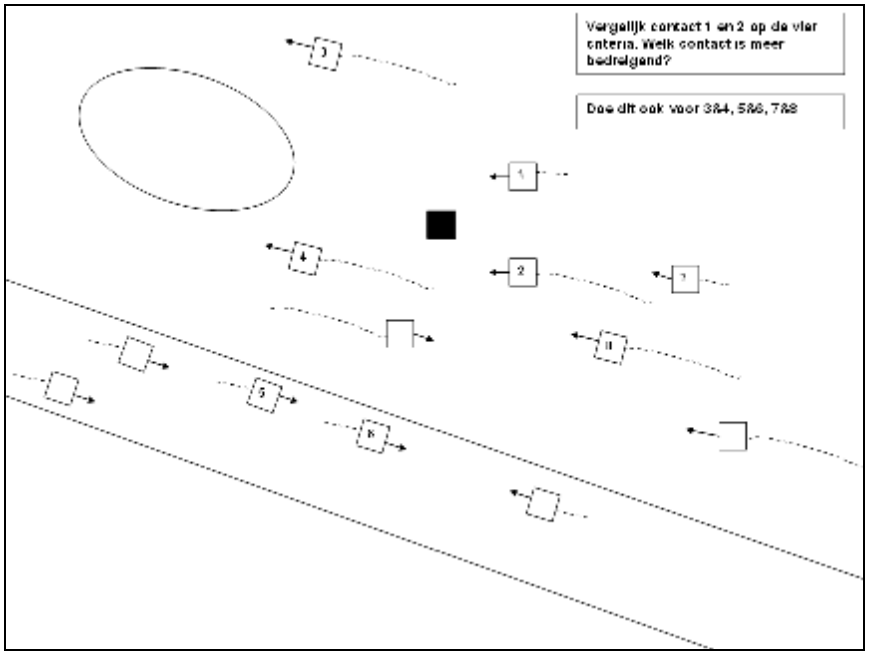
- De **snelheid** van de contacten.
  De snelheid van een contact kan worden bepaald door een contact te volgen. Achter ieder contact is een spoor te zien. Een lang spoor betekent een hoge snelheid, een kort spoor betekent een lage snelheid. Indien een contact een hoge snelheid heeft is dit dreigender dan een contact met een lage snelheid.

- Gebied waar het contact vaart: binnen of buiten de **vaarroute**.
  De vaarroute is een gebied waar het normale scheepsverkeer doorheen vaart, zoals vrachtschepen. De vaarroute wordt weergegeven door 2 lijnen. Contacten buiten de vaarroute zijn daarom dreigender dan contacten die binnen de vaarroute varen.

Om de 5 meest bedreigende contacten te kunnen bepalen moeten de contacten met elkaar worden vergeleken op deze vier criteria. Ga steeds voor elk criterium de mate van bedreiging na. Een contact dat op 4 criteria een hoge bedreigingsscore heeft, is bedreigender dan een contact dat op 2 criteria een hoge bedreigingsscore scoort. De criteria zijn allemaal even belangrijk.

## Voorbeeld

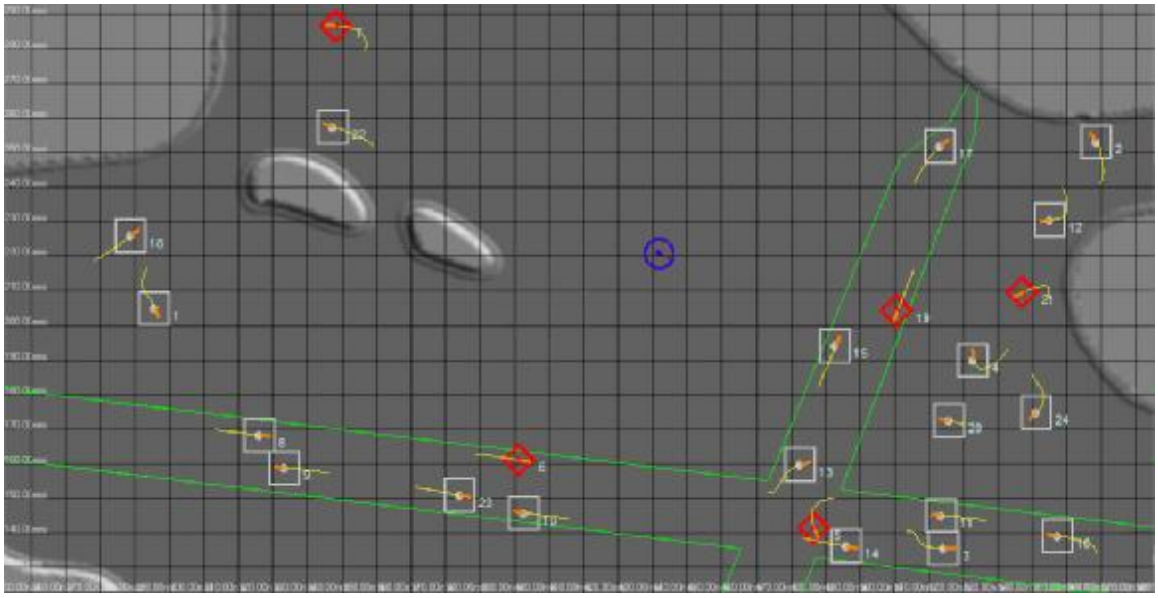




# Appendix E: Participant instructions on the task

U bent commando centrale officier (CCO) op een marine schip en uw taak is om van de schepen (contacten) die in de omgeving varen te bepalen of deze een bedreiging vormen voor uw eigen schip of niet. Straks krijgt u een radarscherm te zien waarop uw eigen schip en de andere contacten zijn weergegeven. Het is de bedoeling dat u op basis van de criteria zo snel mogelijk **de vijf meest bedreigende contacten** identificeert en rood markeert. Echter, welke vijf contacten het meest bedreigend zijn kan veranderen. Het beeld moet dus continu worden aangepast. Om de taak goed te volbrengen is het belangrijk is dat er tijdens de taak niet meer, maar ook niet minder dan vijf contacten rood gemarkeerd zijn. Het onterecht markeren van een contact als een van de vijf meest bedreigende contacten wordt net zo fout gerekend als het niet markeren van een contact dat wel bij de vijf meest bedreigende contacten hoort. U kunt een contact markeren door er met de muis op te klikken. Door op het contact te klikken wordt deze rood, vervolgens bij nog een keer klikken weer wit. Van te voren krijg je de gelegenheid om dit even te oefenen.

Zie figuur 1 voor het scherm van de taak. De symbolen zijn iets anders dan tijdens de toets. De rode contacten zijn contacten die als bedreigend zijn gemarkeerd. De blauwe cirkel is het eigen schip. Elk contact heeft een uniek identificatienummer. De groene banen zijn de vaarroutes. De licht grijze gebieden zijn stukken land. De gele streep achter het contact geeft de snelheid aan, de oranje streep voor het contact de vaarrichting.



Figuur 1: Het radarscherm van de beeldopbouw taak.

# Appendix F: Participant instructions on the support

## *De ondersteuning*

Straks gaan we beginnen met het experiment. Gedurende het experiment zul je geholpen worden met het juist uitvoeren van de taak. Dit wordt gedaan door je aandacht te trekken naar bepaalde, voor de uitvoering van de taak belangrijke, contacten. De aandacht wordt getrokken door contacten feller te laten oplichten, zodat deze extra opvallen. Hieronder staan twee contacten, de linker vereist extra aandacht, de rechter niet.


Figuur 2: Twee uitvergrootte contacten: aandacht dient getrokken te worden (links) of niet (rechts).

Voor de bovengenoemde aandachtssturingsmethode zijn verschillende ondersteunende systemen beschikbaar, waarvan de werking op de achterzijde wordt beschreven. Je hoeft de uitleg hierover pas te lezen vlak voordat je aan elke run gaat beginnen. Voordat je begint met een run wordt steeds duidelijk gemaakt welke vorm van ondersteuning gebruikt gaat worden.

### Geen ondersteuning (NS)

Dit spreekt voor zich. In deze vorm zal geen ondersteuning geboden worden en moet je dus op eigen inzicht de taak zo goed mogelijk uitvoeren.

### Gefixeerde ondersteuning (FS)

De vijf meest bedreigende contacten zullen bij deze ondersteuning oplichten. De computer is echter niet in staat om de bedreigingscore 100% betrouwbaar te meten. Let dus zelf ook goed op, omdat de ondersteuning niet in alle gevallen juist zal zijn.

### Adaptieve ondersteuning (AS)

## Liberaal adaptieve ondersteuning (LAS)

Deze vorm van ondersteuning gebruikt de volgende informatie:

1) Jouw selectie van de vijf meest bedreigende contacten tot nog toe.

De ondersteuning kan hierdoor onderscheid maken tussen geselecteerde en niet-geselecteerde contacten. De niet-geselecteerde contacten met een hoge bedreigingscore zullen helderder worden, omdat deze in aanmerking kunnen komen om geselecteerd te worden. Van de geselecteerde contacten lichten degenen met lage bedreigingscores op, omdat deze in aanmerking kunnen komen om uit de selectie verwijderd te worden.

**Let op:** Als een contact oplicht, betekent dit niet automatisch dat je deze moet selecteren. Er wordt alleen aangegeven dat je dit contact in de gaten moet houden. Daarnaast is bij deze ondersteuning de bedreigingscore niet 100% betrouwbaar.

## Conservatief adaptieve ondersteuning (CAS)

Voor dit type ondersteuning wordt de volgende informatie gebruikt:

1) Jouw selectie van de vijf meest bedreigende contacten tot nog toe.
2) De contacten waarvan de computer denkt dat je er aandacht voor hebt.

Deze vorm van ondersteuning is hetzelfde als de LAS conditie, maar laat contacten niet oplichten als je er al aandacht voor hebt.

**Let op:** Als een contact oplicht, betekent dit niet automatisch dat je deze moet selecteren. Er wordt alleen aangegeven dat je dit contact in de gaten moet houden. Daarnaast is bij deze ondersteuning de bedreigingscore niet 100% betrouwbaar.